EPFL

Thèse n° 10 642

Infusing structured knowledge priors in neural models for sample-efficient symbolic reasoning

Présentée le 23 février 2024

Faculté des sciences et techniques de l’ingénieur
Laboratoire de traitement des signaux 2
Programme doctoral en génie électrique

pour l’obtention du grade de Docteur ès Sciences

par

Mattia ATZENI

École
polytechnique
fédérale
de Lausanne

2024

I am just a child who has never grown up.
I still keep asking this "how" and "why" questions.
Occasionally, I find an answer.
— Stephen Hawking

Acknowledgements

Embarking in my PhD has been much more than a professional pathway, since the very beginning it unfolded into a profound journey and a transformative adventure of personal growth. Reaching this milestone would have not been possible without the support of many people who have played a pivotal role and stood by me throughout the most challenging phases of this journey. Taking a moment for self-reflection, I would like to express my gratitude to everyone who contributed to this enriching experience.

First and foremost, I wish to thank my supervisors, Andreas Loukas and Pierre Vandergheynst. Andreas, I miss the words to express how thankful I am to you. You have surpassed all expectations, being the best supervisor I could be fortunate enough to find. Thanks for your unparalleled support, for being always understanding, and for being a source of inspiration for the duration of my whole PhD. I learned a great deal from you, and, most importantly, your attitude towards research, your passion, and your rigor have made an indelible impact on my professional growth. I would like to extend my appreciation to Pierre for giving me the invaluable opportunity to join his remarkable group at EPFL, thereby jumpstarting my PhD journey.

I am extremely thankful to the remaining members of the jury, Volkan Cevher, James Henderson, Pasquale Minervini, and Rajarshi Das. Thanks for agreeing to be part of the committee, for generously dedicating your time and sharing your expertise.

During my doctoral studies, I have been fortunate to collaborate with several mentors. To start with, I wish to thank Mrinmaya Sachan, for his unwavering mentorship and collaboration from the beginning to the completion of my doctoral studies. Mrinmaya, thanks for hosting me at ETH Zurich, providing invaluable feedback on my work, engaging in the exchange of ideas, and your steadfast support. Similarly, I would like to express my heartfelt gratitude to Jasmina Bogojeska for being the most supportive presence during my time at IBM Research. Your understanding and encouragement have been crucial in a delicate phase of my journey. Thanks for being keen to my interests, I really appreciated your dedication to ensuring that I had a positive experience during my PhD. I am grateful to Keerthiram Murugesan, for granting me the opportunity to join different works and collaborations, for betting on me and giving me the freedom to explore and bring my contributions to these engaging projects. I also wish to thank Nicola Cancedda for hosting me at Meta AI during my internship in

London. Your efforts to facilitate a successful completion of the internship have been truly commendable and made my experience more rewarding and fulfilling.
Besides these mentors, I would like to extend my heartfelt gratitude to the numerous individuals I had the privilege of meeting during my PhD journey, each of whom contributed either directly or indirectly to the work presented in this thesis. I need to thank my coauthors during my time at IBM Research, including Karthik, Pavan, Murray, Gerry, Sadhana, Pushkar, and Kinjal, as well as Lan, Nicolas, Akifumi, Shivam, Declan, Songtao, Michael, and Subhajit. I am grateful to the group I worked with in Zurich, especially Cristiano, Jovan (thanks for introducing me to karting), Andrea, Peter, Mattia, Joana, Anika, Paolo, Katya, Evgeny, Thomas, and Christoph.
I am very grateful to all the amazing students that I met during my doctoral studies, both at EPFL and ETH Zurich. Special thanks to Shehzaad, I really enjoyed working with you. Additionally, I wish to thank all the other students who created a collaborative environment, including Nikos, Giorgos, Shridhar, Shreya, Alessandro, Eleonor, Karolina, Sankalan, Abinav, Jakub, Vilem, Andreas, Zhijing, Tianyu, Volodymyr, and Konstantinos. Similarly, I extend my gratitude to my coworkers at Meta AI, including Simone (thanks for our pool matches), Nora, Mike, Frederic, Louis, Katerina, Kashyap, Despoina, Borislav, and Matteo.
Dulcis in fundo, I need to thank my family, my parents, Maria Pia and Fabrizio, and my brother Gabriele, for their constant support. As a final note, and perhaps somewhat self-indulgently, I want to thank myself as well. This journey demanded persistence, dedicated effort, and a healthy dose of stubbornness. I am grateful to the part of me that persevered, believed in this, and remained focused throughout this challenging yet amazing journey.

Mattia Atzeni

Abstract

The ability to reason, plan and solve highly abstract problems is a hallmark of human intelligence. Recent advancements in artificial intelligence, propelled by deep neural networks, have revolutionized disciplines like computer vision and natural language processing. However, in spite of the amazing progress that we are witnessing, the challenge of creating models that can acquire human-level reasoning abilities sample efficiently persists. To make a step forward, it is crucial to acknowledge that all models inherently carry inductive biases and that human-level intelligence cannot be general and requires the incorporation of appropriate knowledge priors.

Following this chain of thought, this study aims to scrutinize and enhance the reasoning abilities of neural networks by incorporating proper knowledge priors and biasing learning through structured representations. Due to the complexity of the problem at hand, we aim to investigate it through multiple lenses. The thesis unfolds into three main parts, each focusing on distinct tasks and perspectives.

In the first part of the thesis, our research revolves around reasoning and planning in interactive textual environments. We introduce novel environments for evaluating commonsense reasoning skills and decision-making abilities of neural agents. Then, we investigate whether graph-structured representations can serve as appropriate inductive biases for knowledge representation and reasoning with neural agents. We propose agents that use graphs both as a source of prior knowledge and as a model of the state of the world, showing that they act more sample efficiently. Further, we introduce a general algorithm inspired by case-based reasoning to train on-policy agents, improving their planning and out-of-distribution generalization abilities.

In the second part, we isolate core factual reasoning challenges and investigate how language models can reason and benefit from prior knowledge. We delve into language-understanding tasks and introduce an efficient method to navigate large-scale knowledge graphs and answer natural language questions requiring complex logical reasoning and robustness to distributional shifts. Then, we introduce a method to enhance language models with prior knowledge in entity-linking tasks, showing improvements by infusing appropriate structure in the latent space.

Finally, driving inspiration from developmental science, we focus on the core knowl-

edge priors of human intelligence, concentrating our efforts on geometry and topology priors. We introduce a variant of the transformer model that incorporates lattice symmetry priors, showing that it is 2 orders of magnitude more sample efficient than standard transformers on fundamental geometric reasoning.
The contributions of this thesis span several fronts. We achieve state-of-the-art results on several benchmarks, including popular textual environments, standard question answering and entity linking datasets, as well as geometric reasoning tasks. Our text-based neural agents are more sample efficient and resilient to distributional shifts than the baselines. The proposed question answering model is orders of magnitude more scalable than competitive approaches and achieves compositional generalization out of the training distribution. Our entity linking method achieves results comparable to large generative models with 18 times more parameters.

Keywords: Reasoning, Textual reinforcement learning, Text-based games, Commonsense reasoning, Case-based reasoning, Knowledge graphs, Question answering.

Résumé

La capacité de raisonner, de planifier et de résoudre des problèmes très abstraits est une caractéristique de l'intelligence humaine. Les progrès récents de l'intelligence artificielle, propulsés par les réseaux neuronaux profonds, ont révolutionné des disciplines telles que la vision par ordinateur et le traitement du langage naturel. Cependant, malgré les progrès étonnants auxquels nous assistons, le défi de créer des modèles capables d'acquérir efficacement des capacités de raisonnement au niveau humain persiste. Pour faire un pas en avant, il est crucial de reconnaître que tous les modèles comportent intrinsèquement des biais inductifs et que l'intelligence humaine ne peut pas être générale et nécessite l'incorporation de connaissances préalables appropriées.

Suivant cette chaîne de pensée, cette étude vise à examiner et à améliorer les capacités de raisonnement des réseaux de neurones en intégrant des connaissances préalables appropriées et en biaisant l'apprentissage par le biais de représentations structurées. En raison de la complexité du problème en question, nous visons à l'étudier sous plusieurs angles. La thèse se déroule en trois parties principales, chacune se concentrant sur des tâches et des perspectives distinctes.

Dans la première partie de la thèse, nos recherches s'articulent autour du raisonnement et de la planification dans des environnements textuels interactifs. Nous introduisons de nouveaux environnements pour évaluer les capacités de raisonnement de bon sens et les capacités de prise de décision des agents neuronaux. Ensuite, nous étudions si les représentations structurées sous forme de graphes peuvent servir de biais inductifs appropriés pour la représentation des connaissances et le raisonnement avec des agents neuronaux. Nous proposons des agents qui utilisent les graphiques à la fois comme source de connaissances préalables et comme modèle de l'état du monde, démontrant qu'ils agissent de manière plus efficace en échantillonnant. De plus, nous introduisons un algorithme général inspiré du raisonnement basé sur des cas pour former des agents sur la politique, améliorant ainsi leurs capacités de planification et de généralisation hors distribution.

Dans la deuxième partie, nous isolons les principaux défis du raisonnement factuel et étudions comment les modèles linguistiques peuvent raisonner et bénéficier de connaissances antérieures. Nous approfondissons les tâches de compréhension du langage et introduisons une méthode efficace pour naviguer dans des graphiques de connaissances à grande échelle et répondre à des questions en langage naturel nécessitant un raisonnement logique complexe et une robustesse aux changements de distribution. Ensuite, nous introduisons une méthode pour améliorer les modèles de langage avec des connaissances préalables dans les tâches de liaison d'entités, montrant des améliorations en insufflant une structure appropriée dans l'espace latent.

Enfin, en nous inspirant de la science du développement, nous nous concentrons sur les connaissances préalables fondamentales de l'intelligence humaine, en concentrant nos efforts sur les connaissances préalables de la géométrie et de la topologie. Nous introduisons une variante du modèle de transformateur qui intègre les a priori de symétrie de réseau, montrant qu'il est 2 ordres de grandeur plus efficace en matière d'échantillon que les transformateurs standards sur le raisonnement géométrique fondamental.

Les contributions de cette thèse couvrent plusieurs fronts. Nous obtenons des résultats de pointe sur plusieurs benchmarks, notamment des environnements textuels populaires, des réponses aux questions standard et des ensembles de données de liaison d'entités, ainsi que des tâches de raisonnement géométrique. Nos agents neuronaux basés sur du texte sont plus efficaces en matière d'échantillons et plus résilients aux changements de distribution que les lignes de base. Le modèle de réponse aux questions proposé est d'un ordre de grandeur plus évolutif que les approches compétitives et permet une généralisation compositionnelle en dehors de la distribution de la formation. Notre méthode de liaison d'entités permet d'obtenir des résultats comparables aux grands modèles génératifs avec 18 fois plus de paramètres.

Mots clefs : Raisonnement, Apprentissage par renforcement textuel, Jeux textuels, Raisonnement par cas, Knowledge graphs, Question answering.

Sommario

La capacità di ragionare, pianificare e risolvere problemi altamente astratti è un segno distintivo dell'intelligenza umana. I recenti progressi nell'intelligenza artificiale, alimentati dalle reti neurali profonde, hanno rivoluzionato discipline come la visione artificiale e l'elaborazione del linguaggio naturale. Tuttavia, nonostante gli straordinari progressi a cui stiamo assistendo, rimane intatta la sfida di creare modelli in grado di acquisire in modo efficiente capacità di ragionamento di livello umano. Per progredire, è fondamentale riconoscere che tutti i modelli portano intrinsecamente bias induttivi e che l'intelligenza di livello umano non può essere generale e richiede l'incorporazione di adeguate conoscenze a priori.

Seguendo tale catena di pensiero, questo studio mira ad esaminare e migliorare le capacità di ragionamento delle reti neurali incorporando adeguati bias induttivi e influenzando l'apprendimento attraverso rappresentazioni strutturate. A causa della complessità del problema in questione, puntiamo ad indagarlo da punti di vista differenti. A tal fine, la tesi si sviluppa in tre parti principali, ciascuna focalizzata su prospettive e obiettivi distinti.

La prima parte della tesi si concentra su problemi di ragionamento e pianificazione in ambienti testuali interattivi. Introduciamo nuovi ambienti per valutare le capacità di ragionamento e pianificazione degli agenti neurali e investighiamo se le rappresentazioni strutturate a grafo possano servire come bias induttivi appropriati. Proponiamo agenti che utilizzano grafi sia come fonte di conoscenza a priori che come modello dello stato dell'ambiente, dimostrando che agiscono in modo più efficiente. Inoltre, introduciamo un algoritmo generale ispirato al ragionamento basato sui casi per addestrare agenti on-policy, migliorando le loro capacità di pianificazione e generalizzazione fuori dalla distribuzione di training.

Nella seconda parte, isoliamo il problema del ragionamento fattuale e analizziamo le capacità di ragionamento dei language model. Lo studio si concentra su compiti di comprensione del linguaggio e introduciamo un metodo efficiente per esplorare knowledge graph di grandi dimensioni e rispondere a domande in linguaggio naturale che richiedono ragionamenti logici complessi e robustezza a cambiamenti di distribuzione. Inoltre, introduciamo un metodo per migliorare i language model in task di entity linking, infondendo una struttura appropriata nello spazio latente delle

rappresentazioni delle entità.
Infine, traiamo ispirazione dalla psicoligia cognitiva e ci concentriamo sui knowledge prior dell'intelligenza umana, specificamente su prior di geometria e topologia. Introduciamo una variante del trasformer che incorpora i prior del gruppo di simmetria dei lattici, dimostrando che è due ordini di grandezza più efficiente rispetto ai trasformer standard su compiti di ragionamento geometrico.
I contributi di questa tesi spaziano su più fronti. Otteniamo risultati allo stato dell'arte su diversi benchmark, inclusi ambienti testuali, dataset di question answering ed entity linking, nonché task di ragionamento geometrico. I nostri agenti neurali basati su testo sono più efficienti e resilienti a cambiamenti distribuzionali rispetto alle controparti testuali. Il modello di question answering proposto è ordini di grandezza più efficiente degli altri approcci e generalizza al di fuori della distribuzione di training. Il nostro metodo di entity linking raggiunge risultati paragonabili a modelli generativi di grandi dimensioni con 18 volte più parametri.

Parole chiave: Ragionamento, Reinforcement learning testuale, Giochi testuali, Ragionamento basato su casi, Grafi di conoscenza, Question answering.

Zusammenfassung

Die Fähigkeit, hochabstrakte Probleme zu denken, zu planen und zu lösen, ist ein Kennzeichen der menschlichen Intelligenz. Jüngste Fortschritte in der künstlichen Intelligenz, vorangetrieben durch tiefe neuronale Netze, haben Disziplinen wie Computer Vision und natürliche Sprachverarbeitung revolutioniert. Doch trotz der erstaunlichen Fortschritte, die wir erleben, bleibt die Herausforderung bestehen, Modelle zu erstellen, die auf effiziente Weise Denkfähigkeiten auf menschlicher Ebene erwerben können. Um einen Schritt nach vorne zu machen, ist es wichtig anzuerkennen, dass alle Modelle von Natur aus induktive Verzerrungen aufweisen und dass Intelligenz auf menschlicher Ebene nicht allgemein sein kann und die Einbeziehung geeigneter Wissensvorkenntnisse erfordert.

Diesem Gedankengang folgend zielt diese Studie darauf ab, die Denkfähigkeiten neuronaler Netze zu untersuchen und zu verbessern, indem geeignete Wissensvoraussetzungen einbezogen und das Lernen durch strukturierte Darstellungen beeinflusst werden. Aufgrund der Komplexität des vorliegenden Problems ist es unser Ziel, es aus mehreren Blickwinkeln zu untersuchen. Die Arbeit gliedert sich in drei Hauptteile, die sich jeweils auf unterschiedliche Aufgaben und Perspektiven konzentrieren.

Im ersten Teil der Arbeit dreht sich unsere Forschung um das Denken und Planen in interaktiven Textumgebungen. Wir stellen neuartige Umgebungen zur Bewertung der Fähigkeiten des gesunden Menschenverstandes und der Entscheidungsfähigkeit neuronaler Agenten vor. Anschließend untersuchen wir, ob graphstrukturierte Darstellungen als geeignete induktive Verzerrungen für die Wissensdarstellung und das Denken mit neuronalen Agenten dienen können. Wir schlagen Agenten vor, die Graphen sowohl als Quelle für Vorwissen als auch als Modell des Zustands der Welt nutzen und zeigen, dass sie effizienter agieren. Darüber hinaus führen wir einen allgemeinen Algorithmus ein, der auf fallbasiertem Denken basiert, um richtlinienkonforme Agenten zu schulen und ihre Planungs- und Out-of-Distribution-Generalisierungsfähigkeiten zu verbessern.

Im zweiten Teil isolieren wir die zentralen Herausforderungen beim sachlichen Denken und untersuchen, wie Sprachmodelle schlussfolgern und von Vorwissen profitieren können. Wir befassen uns mit Sprachverständnisaufgaben und stellen eine effiziente

Methode zum Navigieren in großen Wissensgraphen und zur Beantwortung natürlichsprachlicher Fragen vor, die komplexes logisches Denken und Robustheit gegenüber Verteilungsverschiebungen erfordern. Anschließend stellen wir eine Methode zur Verbesserung von Sprachmodellen mit Vorkenntnissen in Entitätsverknüpfungsaufgaben vor und zeigen Verbesserungen durch die Einführung einer geeigneten Struktur in den latenten Raum.
Schließlich konzentrieren wir uns, inspiriert von der Entwicklungswissenschaft, auf die Kernwissensprioritäten der menschlichen Intelligenz und konzentrieren unsere Bemühungen auf Geometrie- und Topologieprioritäten. Wir stellen eine Variante des Transformatormodells vor, die Priors der Gittersymmetrie berücksichtigt, und zeigen, dass es aufgrund grundlegender geometrischer Überlegungen zwei Größenordnungen effizienter bei der Stichprobe ist als Standardtransformatoren.
Die Beiträge dieser Arbeit erstrecken sich über mehrere Fronten. Wir erzielen hochmoderne Ergebnisse bei mehreren Benchmarks, darunter gängige Textumgebungen, Datensätze zur Standardfragebeantwortung und Entitätsverknüpfung sowie geometrische Denkaufgaben. Unsere textbasierten neuronalen Agenten sind stichprobeneffizienter und widerstandsfähiger gegenüber Verteilungsverschiebungen als die Basislinien. Das vorgeschlagene Frage-Antwort-Modell ist um Größenordnungen skalierbarer als konkurrierende Ansätze und erreicht eine kompositorische Verallgemeinerung aus der Trainingsverteilung. Unsere Entity-Linking-Methode erzielt mit 18-mal mehr Parametern vergleichbare Ergebnisse wie große generative Modelle.

Stichwörter: Argumentation, Textuelles Verstärkungslernen, Textspiele, Fallbasiertes Denken, Wissensgraphen, Beantwortung von Fragen.

Contents

1 Introduction

1.1 A surprisingly old debate

With the dawn of our current era of deep learning and neural networks, the pace and impact of new advances in the field of Artificial Intelligence (AI) have become astounding. Deep neural networks have fueled a remarkable progress in an extremely wide range of tasks, leading to dramatic improvements in broad research areas like computer vision and natural language understanding (Krizhevsky et al. 2012; LeCun et al. 2015; Mikolov et al. 2013; Vaswani et al. 2017; Devlin et al. 2019; OpenAI 2023). Nevertheless, although neural networks are extremely suited to pattern recognition, classification, and perception tasks, the design of models that can deliberate, think, and reason using knowledge still opens the path to several research challenges and opportunities (Bengio 2017; Marcus 2020; Marcus 2018). Indeed, problems that involve reasoning operations, such as deduction, induction, abduction, spatial or temporal reasoning, and probabilistic inference are still difficult to solve with neural networks (Lippi 2016; Atzeni, Sachan, et al. 2023; Kassner, Krojer, et al. 2020). Even in light of the amazing progress that we have witnessed with large language models (LLMs) (OpenAI 2023; Radford et al. 2019; Brown et al. 2020), the ability to acquire reasoning skills *sample efficiently* remains one of their main limitations, as they may "hallucinate" facts and make reasoning mistakes (OpenAI 2023; Kassner, Krojer, et al. 2020).

Reasoning problems are often better solved with search algorithms and traditional symbolic approaches (Muggleton et al. 1994; Cropper et al. 2019; Marcus 2020). The challenge of solving them with deep learning has recently prompted an intriguing connection between the current status of AI research and theories in cognitive psychology. In particular, recently, the theory on fast and slow thinking proposed by Kahneman (2011) has been regarded to have fascinating connections with AI research (Bengio 2019; Booch et al. 2021; Ganapini et al. 2022). Kahneman's work postulates the existence of two different modes of thinking and decision-making processes that occur in the human mind. These two modes are termed *System 1* and *System 2* respectively. *System 1* thinking is fast, intuitive, and automatic. It operates effortlessly and quickly, often relying on heuristics or mental shortcuts to make decisions. This mode of thinking is efficient for everyday tasks such as recognizing faces, reading emotions, and navigating familiar environments. However, it can also lead to cognitive biases and errors because it relies on mental shortcuts rather than deep analysis. On the other hand, *System 2* thinking is slow, deliberate, and analytical. It requires critical thinking, problem-solving, and rational decision-making. This mode of thinking is used for complex tasks, such as solving mathematical problems or evaluating evidence in a logical and systematic manner.

Deep learning models, despite their remarkable success, often exhibit a dominance of *System 1*-like behavior, excelling in pattern recognition but struggling with systematic symbolic reasoning. This connection has spawned several debates in the AI community on how to reach models with better reasoning abilities. These debates can be summarized into two main lines of thoughts, one arguing that symbolic reasoning can be achieved with an approach purely based on deep learning and neural networks (Bengio 2019; Bengio 2017; A. Goyal et al. 2022), and another one that advocates for hybrid approaches between neural networks and symbolic methods (Marcus 2020; Marcus 2018).

At the root of this debate, we can distinguish two main approaches to AI (Broeck 2019). The first is an *inductive* vision, which:

> *approaches the problem of designing intelligent machines by postulating a large number of very simple information processing elements, arranged in a* [...] *network, and certain processes for facilitating or inhibiting their activity.* (Feigenbaum and Feldman)

The latter, on the other hand, is a *deductive* vision, which postulates that:

> *intelligent performance by a machine is an end difficult enough to achieve starting from scratch, and so [it requires building into the] systems as much complexity of information processing as [...] able to understand and communicate to a computer.* (Feigenbaum and Feldman)

From the definitions above, we can see that the inductive approach reminds of *deep learning*, whereas the deductive one resembles *symbolic methods*. Despite being seemingly a very recent distinction, the quotes above are from the book of Feigenbaum et al. (1963), which dates back to 60 years ago. Clearly, Feigenbaum et al. (1963) did not use the terms that are popular today, but referred to the inductive and deductive approaches as *neural cybernetics* and *cognitive model builders* respectively. Thus, we see that this debate that we have recently experienced in AI research has actually shaped the field since its early stages.

We could argue that the roots of this debate extend even further into the past, as we can trace it back to the age-old philosophical distinction between nature and nurture. The question of whether intelligence is innate (nature) or it is acquired from environmental factors (nurture) has been a source of fascination and controversy throughout the history of human thought. Contemporary perspectives recognize the intertwined nature of these factors, emphasizing their complex interplay in understanding human development and cognition (Spelke et al. 2007).

Embracing this modern vision, in this thesis we aim to investigate the hypothesis that neural networks can learn symbolic reasoning tasks sample efficiently, by introducing proper knowledge priors and inductive biases, in the form of architectural assumptions, to guide learning towards specific, structured hierarchical representations and computations. To verify this hypothesis, we focus on several tasks, as detailed in Section 1.3. In the remainder of this chapter, we explain how neural networks can benefit from knowledge priors and we provide the main motivation and a summary of the work described in this thesis.

1.2 The need for knowledge priors in deep learning

1.2.1 Some limitations of deep learning today

Despite the remarkable achievements that the AI community has witnessed with the advent of deep learning, it is crucial to acknowledge the limitations that persist in the field. Understanding these limitations is essential for both researchers and practitioners, as it informs the responsible and effective application of deep learning technologies and it provides guidance for further research.

The aim of this thesis, is to research methods for learning complex tasks involving symbolic reasoning with neural networks. As we have already mentioned, deep learning approaches struggle with this kind of tasks. To be more precise, however, two main limitations of current deep learning models are mostly relevant.

Out-of-distribution generalization. Many recent works have highlighted the challenges faced by deep learning in dealing with distributional shifts between the training set and the test set. We refer to this setting as out-of-distribution (OOD) generalization, as we aim to generalize out of the training distribution. This is particularly important in symbolic reasoning tasks, as we expect the model to be able to learn symbolic rules that generalize irrespectively of the distribuion of atomic symbols in the training set. Conventional supervised learning, which relies on *independent and identically* *distributed* (i.i.d.) data, is usually inadequate for addressing OOD generalization, as distributional shifts undermine this foundational assumption (Lake, Ullman, et al. 2017; Lake and Baroni 2018; Bahdanau, Murty, et al. 2019). Therefore, the development of algorithms capable of robust OOD generalization has emerged as a key research focus, encompassing disentangled representation learning (T. Wang et al. 2021; Locatello et al. 2019), causal representation learning (Schölkopf et al. 2021; M. Yang et al. 2021), and optimization techniques (Sagawa et al. 2019). Finally, evaluating OOD generalization performance presents significant hurdles, which fueled the cretion of datasets and unique evaluation metrics distinct from traditional i.i.d. benchmarks (Lake and Baroni 2018; Keysers et al. 2020). In this thesis, we introduce a dataset to measure OOD generalization in the context of textual reinforcement learning (Chapter 2) and we evaluate methods on this and other benchmarks in Chapters 2, 3, and 4. In Chapter 5, we evaluate OOD generalization on the dataset of Keysers et al. (2020).

Sample efficiency. Neural networks, particularly deep ones, often require vast datasets to generalize well and make accurate predictions. When data is scarce, these models may struggle to learn meaningful patterns and can easily overfit, memorizing the training data rather than generalizing to unseen examples. This limitation is especially significant in domains where collecting data is expensive or time-consuming, hindering the practicality of deploying deep learning techniques. Learning from few examples is an active area of research today, to make these models more applicable in data-constrained scenarios (Vinyals et al. 2016; Snell et al. 2017; Prabhudesai et al. 2021). Sample efficiency is particularly relevant in reasoning tasks, as we expect the model to be able to infer generalizable rules from few examples. As we will detail later, sample efficiency is a recurrent theme in this thesis. In Chapters 2, 3, and 6, we use prior knowledge to improve the sample efficiency. In Chapter 4, we introduce a reinforcement learning algorithm inspired by case-based reasoning to improve sample efficiency and generalization performance of the models. Finally, Chapter 7 focuses on few-shot learning of geometric reasoning tasks, as we aim to infer a reasoning chain from a very limited number of examples.

1.2.2 Infusing knowledge priors in neural networks

In order to move towards AI models with human-level reasoning abilities, it is essential to address the limitations above. Several recent works advocate for the need of infusing proper knowledge priors and inductive biases in neural networks (Battaglia et al. 2018; Bengio 2017; Bengio, Deleu, et al. 2020; Hudson et al. 2018; Lake, Ullman, et al. 2017; Lake and Baroni 2018). After all, by the "*no free lunch*" theorem of machine learning, every model is biased and will favor some hypothesis class over the others. Therefore, we could argue that there is no completely general intelligence and some knowledge priors or inductive biases are needed. It becomes then a crucial problem to identify proper biases and knowledge priors to guide the learning process.

An early approach to guide learning towards more structured computations was the introduction of neural networks augmented with an external memory. The main intuition was that this could allow the model to efficiently reason over variables and represent structured data over long time scales. Among the initial contributions in this field, Neural Turing Machines (NTM) (Graves, Wayne, and Danihelka 2014) and Memory Networks (Weston et al. 2015) had great success and have been widely employed. Other examples of memory-augmented architectures include Dynamic Memory Networks (Kumar et al. 2016), End-to-end Memory Networks (Sukhbaatar et al. 2015), and the Differentiable Neural Computer (DNC) (Graves, Wayne, Reynolds, et al. 2016). More recently, Csordas et al. (2019) also proposed some interesting improvements to the DNC.

Recently, the use of graph neural networks (GNN) (Kipf et al. 2017; Gori et al. 2005) and graph-structured data has also been a popular approach to bias the learning process with structured information. Although the idea of developing and studying neural networks capable of manipulating and natively processing graph-structured inputs dates back to more than ten years ago (Gori et al. 2005; Scarselli et al. 2009), this kind of models has recently discovered and gained an unprecedented and wider popularity on several different tasks and domains (Battaglia et al. 2018). Such architectures have been proved useful for many problems that require reasoning on a set of discrete entities, such as combinatorial optimization (Khalil et al. 2017) and satisfiability (Selsam et al. 2018). More recently, Kool et al. (2019) successfully applied a model based on the Transformer architecture (Vaswani et al. 2017) to solve routing problems on graphs, like the well-known Traveling Salesman Problem (TSP). Some of the most interesting and promising contributions in this field include message-passing neural networks (Gilmer et al. 2017) and non-local neural networks (X. Wang et al. 2018). Also, the graph networks, that have been introduced by Battaglia et al. (2018), generalize several previous works, thereby providing a fundamental building block for applying deep learning to structured knowledge.

Human-level capabilities like abstract symbolic reasoning require the possibility of compositionally constructing new inferences and behaviors from known building blocks. Therefore, recent research advocated that compositionality should also be incorporated directly into the architecture of the models (Battaglia et al. 2018). Several lines of research are indeed addressing these issues by leveraging the use of proper inductive biases towards more structured cognitive models (Bengio 2017; Bengio, Deleu, et al. 2020; Hudson et al. 2018; Bahdanau, Murty, et al. 2019).

1.2.3 Knowledge priors and human intelligence

Research in cognitive psychology has also postulated the idea that human intelligence is based on some core knowledge priors (Spelke et al. 2007). The *Core Knowledge* theory challenges two prevailing views of human cognition. One view posits that the human mind is a flexible and general learning system, capable of adapting to various experiences. The opposing view suggests that the human mind comprises specialized mechanisms, each evolved to serve specific functions. Instead, the *Core Knowledge* theory proposes that humans possess a limited number of distinct systems of core knowledge, which form the basis for cognitive development. These core knowledge systems are foundational and serve as the building blocks for more advanced cognitive abilities and fall into four main categories.

- **Objectness priors**: this system enables the representation of inanimate objects and their mechanical interactions, such as understanding the principles of object permanence and gravity.
- **Agentness and goal-directedness priors**: it involves the recognition of agents and their goal-directed actions, allowing individuals to comprehend and predict the behavior of others.
- **Numerical priors**: this system pertains to numerical relationships, including ordering, addition, and subtraction, providing a fundamental sense of quantity and numerical concepts.
- **Geometry and topology priors**: humans possess an innate understanding of places in the spatial layout and their geometric relationships, allowing for navigation and spatial reasoning.

The *Core Knowledge* theory posits that these foundational systems serve as the basis for the development of more complex cognitive skills and belief systems, challenging the dichotomy between a single general-purpose learning system and numerous specialized mechanisms in the study of human cognition. Recently, this theory has inspired the development of AI benchmarks, like the Abstraction and Reasoning Corpus (ARC) (Chollet 2019). In Chapter 7, we draw inspiration from this theory and we propose a method to infuse a category of *Core Knowledge* priors (geometry and topology priors) in transformer models.

1.3 A problem that requires multiple perspectives

Enhancing neural networks with the capacity for efficient reasoning over structured data is a complex problem that is difficult to formalize precisely, as it encompasses several applications and tasks. Therefore, we believe that the complexity of the problem necessitates the development of approaches that draw from multiple perspectives and applications. To comprehensively explore this multifaceted research challenge, we turn our attention to a series of distinct yet interconnected tasks.

1.3.1 Symbolic reasoning in interactive environments

Interactive environments serve as a fertile ground for investigating complex reasoning problems, offering a unique platform to explore a multitude of challenges that arise from long-term planning and intricate decision-making processes. Moreover, reinforcement learning problems are generally well suited for researching neural models that are robust to the limitations mentioned in Section 1.2.1. Indeed, agents in interactive environments often face several challenges, including:

- **changes in distribution**, due to either their own actions or the actions of the other agents;
- **partial observability**, meaning that they only have access to observations that might not provide a complete representation of the state of the environment;
- **combinatorial action spaces**, as the structure of the action space can become combinatorially large in some settings, depending on the problem and the number of agents;
- **long-term credit assignment**, as actions taken in the past can turn useful several time steps later in the future.

Experiencing distributional shifts implies that the model needs to generalize robustly out of the training distribution. The problems of long-term credit assignment and combinatorial action spaces result in high sample complexity, hence the agents need to be able to learn a generalizable policy sample efficiently.

Within the realm of interactive environments, we have chosen to concentrate our efforts on textual environments, namely environments where observations, actions and feedback are expressed in the form of natural language text. In this context, text-based games have recently emerged as a promising framework to drive advances in RL research (Côté et al. 2018; M. Hausknecht et al. 2020). Indeed, this setting provides a rich use case for studying grounded language learning and how information from text can be utilized in sequential planning and decision making (Ammanabrolu and M. J. Hausknecht 2020; Narasimhan et al. 2015; Zahavy et al. 2018). Prior work has explored text-based RL on textual games originally designed for humans, like multi-user dungeon games (Narasimhan et al. 2015) and text-adventure games like *Zork* (Zahavy et al. 2018). Recently, research has also shifted towards games explicitly designed for machine-learning agents (Adolphs et al. 2019), as artificial games can have well-defined goals and controllable complexity. As an example, Côté et al. (2018) introduced `TextWorld`, a sandbox learning environment for training and evaluating RL agents on text-based games.

This task allows us to assess several hypotheses and investigate different research questions. First, we aim to understand whether using prior structured knowledge can improve the sample efficiency and the generalization capabilities of a neural agent based on a large language model. We assess this hypothesis in Chapter 2, using commonsense knowledge extracted from a knowledge graph (Speer et al. 2017). Then, in Chapter 3, we verify whether the use of graph-based representations can provide a helpful inductive bias for reasoning and planning operations. Finally, we explore reasoning approaches for sample-efficient reinforcement learning by proposing an algorithm inspired by case-based reasoning in Chapter 4. We provide an outline of the thesis and more details in Section 1.4.

1.3.2 Factual reasoning in static contexts

Though interactive environments enable us to explore the capabilities of neural networks for reasoning and planning in complex scenarios, this task severely restricts the amount of factual knowledge that the model can leverage and reason upon, as the environments feature a limited number of domain-specific entities. Therefore, our research focuses also on a different set of tasks. More precisely, we aim to investigate the ability of neural networks and language models to reason over prior factual knowledge provided to the model. When combined with the insights gained from interactive environments, this additional perspective contributes to a more comprehensive understanding of reasoning capabilities within neural networks.

To probe the ability of language models and neural networks to perform factual reasoning, we focus on language understanding tasks and provide the model with access to factual information in the form of a knowledge graph (KG). We aim to investigate two main properties of neural networks:

- the ability of the model to reason over the information in the KG efficiently and generalize compositionally out of the training distribution;
- the extent to which prior knowledge can improve performance of the model on language understanding tasks, both in terms of generalization and efficiency.

For the former property, we consider the task of knowledge-based question answering (KBQA), namely answering natural language questions over a knowledge base. This task allows us to probe the ability of the model to access factual information efficiently and perform logical deductions over the retrieved knowledge, even when the knowledge graph comprises millions of edges. Additionally, it allows us to investigate the ability of the model to generalize to reasoning patterns robustly, even when the distribution of questions in the training set is different from the distribution in the test set (Keysers et al. 2020).

On the other hand, to investigate whether language models can benefit from prior knowledge, we consider an entity linking task. Entity linking is a well-known task in natural language processing (NLP) and involves the disambiguation and identification of entities mentioned within textual data. It has been shown that prior knowledge of the type of a mention can greatly improve downstream performance (Raiman et al. 2018). While language models should already have this kind of knowledge (Aghajanyan et al. 2022) and be able to achieve high performance on entity disambiguation tasks (state-of-the-art performance is generally achieved by large generative models), our goal is to investigate whether reasoning over prior structured knowledge can improve their efficiency and the downstream performance of smaller models. Therefore, we will use a knowledge graph without explicit type labels to provide additional information to the language model. The model then has to reason over prior knowledge to infer the type of the mention and link it to the correct entity.

1.3.3 Geometric reasoning in abstract symbolic tasks

The two lines of work described above aim at assessing and enhancing the reasoning capabilities of neural networks by using knowledge graphs as either an inductive bias or a source of prior knowledge. In our pursuit of infusing knowledge priors in neural networks, however, a key research question is what kind of priors should be infused in the model to improve reasoning abilities, sample efficiency and generalization (Battaglia et al. 2018).

As mentioned in Section 1.2.3, the *Core Knowledge* priors for human intelligence have been studied extensively in developmental science (Spelke et al. 2007), following the theory that humans are endowed with a small number of separable systems of core knowledge, so that new flexible skills and belief systems can build on these core foundations. Recent research in artificial intelligence (AI) has postulated the idea that the same priors should be incorporated in AI systems (Chollet 2019), but it is an open question how to incorporate these priors in neural networks.

Following this chain of thought, the Abstraction and Reasoning Corpus (ARC) (Chollet 2019) was proposed as an AI benchmark built on top of the *Core Knowledge* priors from developmental science. The dataset contains a collection of synthetic tasks, where for each task, the model needs to learn a reasoning chain from a small number of input-output pairs (3.3 input-output pairs on average for each task). Chollet (2019) posits that infusing this kind of priors into neural models is a challenging first step towards human-level AI and that "*solving this specific subproblem is critical to general AI progress*". Further, he argues that ARC "*cannot be meaningfully approached by current machine learning techniques, including Deep Learning*".

As the problem of infusing all *Core Knowledge* priors in neural networks is remarkably complex, we focus on one category, specifically geometry and topology priors. This allows us to draw connections to other lines of research that have proposed methods to incorporate geometric priors in neural networks by rendering them invariant (or equivariant) to specific transformations (Bronstein et al. 2021). However, as we focus on geometric reasoning, in this thesis, we rather aim to design models that can learn functions involving geometric transformations of their input efficiently, rather than being oblivious to such transformations. As each task in ARC is defined in terms of a very small number of input-output pairs (3.3 on average), this problem allows us to study and enhance the sample efficiency of neural networks when learning fundamental geometric transformations.

1.4 Thesis outline

The thesis is organized into three main parts, each addressing one of the three perspectives described in Section 1.3. Each part is structured as described below.

Part I: Symbolic and commonsense reasoning in textual environments

Part I endeavors to address the research path outlined in Section 1.3.1, dealing with symbolic and commonsense reasoning in interactive textual environments.

Chapter 2 examines the problem of infusing RL agents with commonsense knowledge in the context of textual environments requiring commonsense and symbolic reasoning. Such knowledge would allow agents to efficiently act in the world by pruning out implausible actions, and perform look-ahead planning to determine how current actions might affect future world states. We design a new text-based gaming environment called TextWorld Commonsense (TWC) for training and evaluating RL agents with a specific kind of commonsense knowledge about objects, their attributes, and affordances. We also introduce several baseline RL agents which track the sequential context and dynamically retrieve the relevant commonsense knowledge from ConceptNet (Speer et al. 2017). We show that agents which incorporate commonsense knowledge in TWC perform better, while acting more efficiently. We conduct user-studies to estimate human performance on TWC and show that there is ample room for future improvement, confirming that our benchmark is suitable to drive further research in this area.

Chapter 3 builds on the work described in Chapter 2 and proposes an agent that models that state of the world using a graph-based representation (similar to the work of Ammanabrolu and M. J. Hausknecht (2020) and Adhikari et al. (2020)) and chooses its own actions by reasoning jointly using prior knowledge and such a representation of state of the game. We show that agents that reason jointly on the state graph and commonsense knowledge graph outperform baseline agents while being more sample efficient, confirming our hypothesis that knowledge graphs are good inductive biases for knowledge representation and that prior and local knowledge are complementary.

Finally, Chapter 4 reuses the same model of Chapter 3, but we propose a general method inspired by case-based reasoning to train agents and generalize out of the training distribution. The case-based reasoner collects instances of positive experiences from the agent's interaction with the world in the past and later reuses the collected experiences to act efficiently. We show that the proposed approach consistently improves existing methods, obtains good out-of-distribution generalization, and achieves new state-of-the-art results on widely used environments.

Part II: Factual reasoning in language-understanding tasks

In this part, we shift our attention to the ability of neural networks and language models to reason over factual knowledge, following the motivation outlined in Section 1.3.2. As mentioned above, we use two well-known language-understanding tasks to probe the ability of language models and neural networks for factual reasoning.

Chapter 5 focuses on question answering and reasoning over knowledge graphs, exploring a method for reasoning over factual knowledge efficiently while using a language model to produce question representations. State-of-the-art approaches to reasoning over KGs usually scale with the number of edges and can only be applied effectively on small instance-dependent subgraphs (H. Sun, Dhingra, et al. 2018; H. Sun, Bedrax-Weiss, et al. 2019; Saxena et al. 2020). We address this issue by showing that multi-hop and more complex logical reasoning can be accomplished separately without losing expressive power. Motivated by this insight, we propose an approach to multi-hop reasoning that scales linearly with the number of relation types in the graph, which is usually significantly smaller than the number of edges or nodes. This produces a set of candidate solutions for more complex reasoning problems, that can be provably refined to recover the original solution. Our experiments on knowledge-based question answering show that our approach solves the multi-hop MetaQA dataset (Zhang et al. 2018), achieves a new state-of-the-art on the more challenging WebQuestionsSP (Yih et al. 2015), is orders of magnitude more scalable than competitive approaches, and can achieve compositional generalization out of the training distribution on the benchmark of Keysers et al. (2020).

Chapter 6 proposes an approach to infusing prior factual knowledge in entity linking methods based on dense retrieval. Concretely, following Raiman et al. (2018), we introduce an method for infusing structural information in the space of entity representations, using prior knowledge of entity types. Inspired by *duck typing* in programming languages, we define the type of an entity based on its relations with other entities in a knowledge graph. Then, porting the concept of box embeddings (Vilnis et al. 2018;

Dasgupta et al. 2020; Abboud et al. 2020) to spherical polar coordinates, we represent relations as boxes on the hypersphere. We optimize the model to place entities inside the boxes corresponding to their relations, thereby clustering together entities of similar type. Our experiments show that our method sets new state-of-the-art results on standard entity-disambiguation benchmarks. It improves the performance of the model by up to 7.9 F_1 points, outperforms other type-aware approaches, and matches the results of generative models with 18 times more parameters.

Part III: Geometric reasoning in abstract symbolic tasks

Finally, Part III focuses on sample-efficient geometric reasoning, as anticipated in Section 1.3.3. As mentioned in Section 1.3.3, the Abstraction and Reasoning Corpus (ARC) (Chollet 2019) and its most recent language-complete instantiation (LARC) (Acquaviva et al. 2021) have been postulated as an important step towards general AI, as they require models that build on fundamental knowledge priors. Yet, even state-of-the-art machine learning models struggle to achieve meaningful performance on these problems, falling behind non-learning based approaches.

To address this issue, Chapter 7 focuses on geometry priors and introduces LATFORMER, a model that incorporates lattice symmetry priors in attention masks. We show that, for any transformation of the hypercubic lattice, there exists a binary attention mask that implements that group action. Furthermore, we show that these masks are the Kronecker product of convolutions of the identity. Hence, our study motivates a modification to the standard attention mechanism, where attention weights are scaled using soft masks generated by a convolutional neural network. Experiments on synthetic geometric reasoning show that LATFORMER requires 2 orders of magnitude fewer data than standard attention and transformers to learn geometric transformations. Moreover, our results on ARC and LARC tasks provide preliminary evidence that these complex datasets do not lie out of the reach of deep learning models.

1.5 List of contributions

The work described in this thesis resulted in the following publications.

- **Chapter 2**: Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. (2021), *Text-based RL Agents with Commonsense Knowledge: New Challenges, Environments and Baselines*, 35th AAAI Conference on Artificial Intelligence, **AAAI 2021**.
- **Chapter 3**: Keerthiram Murugesan, Atzeni, Kapanipathi, Talamadupula, et al. (2021), *Efficient Text-based Reinforcement Learning by Jointly Leveraging State and Commonsense Graph Representations*, Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, **ACL 2021**.

- **Chapter 4**: Atzeni, S. Z. Dhuliawala, et al. (2022), *Case-based Reasoning for Better Generalization in Textual Reinforcement Learning*, International Conference on Learning Representations, **ICLR 2022**.
- **Chapter 5**: Atzeni, Bogojeska, et al. (2021), *SQALER: Scaling Question Answering by Decoupling Multi-Hop and Logical Reasoning*, Advances in Neural Information Processing Systems, **NeurIPS 2021**.
- **Chapter 6**: Atzeni, Plekhanov, et al. (2023), *Polar Ducks and Where to Find Them: Enhancing Entity Linking with Duck Typing and Polar Box Embeddings*, Proceedings the 2023 Conference on Empirical Methods in Natural Language Processing, **EMNLP 2023**.
- **Chapter 7**: Atzeni, Sachan, et al. (2023), *Infusing Lattice Symmetry Priors in Attention Mechanisms for Sample-Efficient Abstract Geometric Reasoning*, International Conference on Machine Learning, **ICML 2023**.

During my doctoral studies, I also co-authored the following papers, which are not included in this document.

- Hoang et al. (2022), *SCERL: A Benchmark for Intersecting Language and Safe Reinforcement Learning*, Second Language and Reinforcement Learning Workshop (LaReL) at NeurIPS 2022.
- Basu et al. (2022), *A Hybrid Neuro-Symbolic approach for Text-Based Games using Inductive Logic Programming*, Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations, CLeaR workshop at AAAI 2022.
- Krivosheev, Atzeni, Mirylenka, Scotton, Miksovic, et al. (2021), *Business Entity Matching with Siamese Graph Convolutional Networks*, The 35th AAAI Conference on Artificial Intelligence, AAAI 2021.
- Krivosheev, Atzeni, Mirylenka, Scotton, and Casati (2021), *Siamese Graph Convolutional Networks for Data Integration*, The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD 2021.

Part I

Symbolic and commonsense reasoning in textual environments

2 Prior structured knowledge for reasoning in interactive environments

2.1 Introduction

Enhancing machine learning models with the ability to perform symbolic reasoning is regarded as a key step to reach systematic generalization and improved sample efficiency in complex tasks (Marcus 2020; Battaglia et al. 2018; Lake and Baroni 2018; Marcus 2018). In particular, reasoning in interactive, partially-observable environments poses several challenges, as learning agents need to perform look-ahead planning on how action trajectories will affect future world states.

Interactive contexts and reinforcement learning (RL) settings provide an ideal use case for studying these challenges, as agents often experience shifts in distribution either due to their own actions or to the actions of other agents. Furthermore, this setting involves planning and systematically generalizing in environments that require reasoning skills and sequential decision making. In this area, text-based games (TBG) (Narasimhan et al. 2015; Côté et al. 2018; Ammanabrolu and Riedl 2019) are a task that is quickly gaining attention, as agents must interact with an external environment using only the modality of text. These games provide several additional challenges, including partial observability, sparse rewards, long-term dependencies and combinatorial action spaces. Moreover, agents need to build a model of the state of the environment from natural language observations, so the best performing agents still struggle to achieve meaningful performance on TBGs.

Research on TBGs has used as benchmarks both text-adventure games originally intended for human players (M. Hausknecht et al. 2020; Ammanabrolu and M. J. Hausknecht 2020) and games specifically designed for research purposes (Côté et al. 2018). However, the former category includes difficult games where the goal is not well-defined whereas the latter comprises simple games that severely restrict the amount of reasoning and commonsense knowledge that the agent needs.

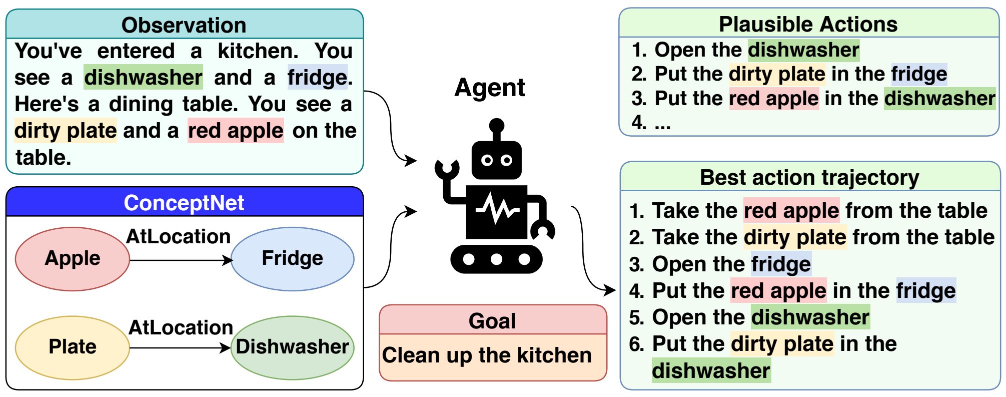

Observation
You've entered a kitchen. You see a dishwasher and a fridge. Here's a dining table. You see a dirty plate and a red apple on the table.
ConceptNet
Apple
AtLocation
Fridge
Plate
AtLocation
Dishwasher
Agent
Goal
Clean up the kitchen
Plausible Actions
1. Open the dishwasher
2. Put the dirty plate in the fridge
3. Put the red apple in the dishwasher
4. ...
Best action trajectory
1. Take the red apple from the table
2. Take the dirty plate from the table
3. Open the fridge
4. Put the red apple in the fridge
5. Open the dishwasher
6. Put the dirty plate in the dishwasher

Figure 2.1: Illustration of a TWC game. The agent is given an initial observation (top left) and has to produce the list of actions (bottom right) that are necessary to achieve the goal (bottom center) using relevant commonsense knowledge from ConceptNet (bottom left).

In order to address these issues, the work described in this chapter builds on the hypothesis that biasing learning with the use of symbolic graph structured representations, like knowledge graphs (KGs), could improve both sample efficiency and generalization performance. In particular, the contributions of the work described in this chapter are as follows.

- We introduce a novel text-based RL environment, called TextWorld Commonsense (or TWC), that requires agents to perform commonsense reasoning to achieve the goal. Achieving goals in this environment requires commonsense knowledge about objects, their properties, locations, and affordances. Efficient use of commonsense knowledge would improve sample efficiency by reducing exploration. Moreover, it would help the agent to perform look-ahead planning and determine how current actions might affect future world states.
- We propose an approach that allows the agent to perform commonsense reasoning over an external knowledge graph. Concretely, we rely on ConceptNet (Speer et al. 2017) as the source of prior knowledge and design a framework where the agent can combine textual information with prior commonsense knowledge dynamically retrieved from this KG.

Figure 2.1 shows a high-level view of both the approach and the novel text-based environments for commonsense reasoning.

In this chapter, we describe the creation and evaluation of TWC. Our experimental results point out that agents with access to the KG generalize better and more efficiently than their text-only counterparts. Moreover, we notice a pronounced gap in performance between automated agents and humans, showing that TWC provides a challenging test-bed for RL agents and can act as a spur to further research in this area.

2.2 TextWorld Commonsense (TWC)

Existing text-based games (Adhikari et al. 2020; Côté et al. 2018) severely restrict the amount and variety of commonsense knowledge that an agent needs to know and exploit. Solving these tasks mostly requires local knowledge about the environment and the involved commonsense reasoning is usually limited and easy to learn directly from the games. Thus, this section proposes a new domain, TextWorld Commonsense (TWC), which allows generating text-based environments where agents can take advantage from performing commonsense reasoning on external KGs.

2.2.1 Constructing TWC

The novel domain introduced in this report, TextWorld Commonsense, enables generating text-based games where the high-level goal is always to tidy up a house by placing objects in their commonsensical locations, as in the example depicted in Figure 2.1. TWC consists of mainly two components:

- a dataset that pairs a set of entities to commonsensical properties of these objects like their usual locations;
- a framework that, based on the TWC dataset, allows generating text-based environments of different difficulty levels to train and evaluate RL agents.

TWC dataset. The TWC environments were generated based on a novel ad-hoc dataset, in order to avoid biasing the games towards any existing KB. A set of vocabulary terms was collected from public sources of information that are orthogonal to commonly used KGs like ConceptNet. These terms were then manually inspected and aggregated in order to build a dataset with several kinds of objects that are typically found in a house environment. The initial set of unique objects was then augmented by adding attributes that may or may not change their target location, as shown in the examples in Table 2.1. The resulting dataset includes a total of 8 room types and more than 900 entities, which can be grouped in three categories: objects, supporters, and containers. Objects are entities that can be carried by the agent, whereas supporters and containers are locations where those objects can be placed. The dataset includes triples of the form $\langle \texttt{o}, \texttt{r}, \texttt{l} \rangle$, whenever it is deemed commonsensical to have an object `o` within room `r` on the supporter/container `l`.

TWC environments. The TWC dataset has been used to generate a novel set of text-based environments requiring commonsense reasoning. The TextWorld (Côté et al. 2018) engine was employed to generate the natural language observations, and the environments were grouped into three difficulty levels (*easy*, *medium*, and *hard*) depending on the total number of objects and rooms in the game. Table 2.2 provides

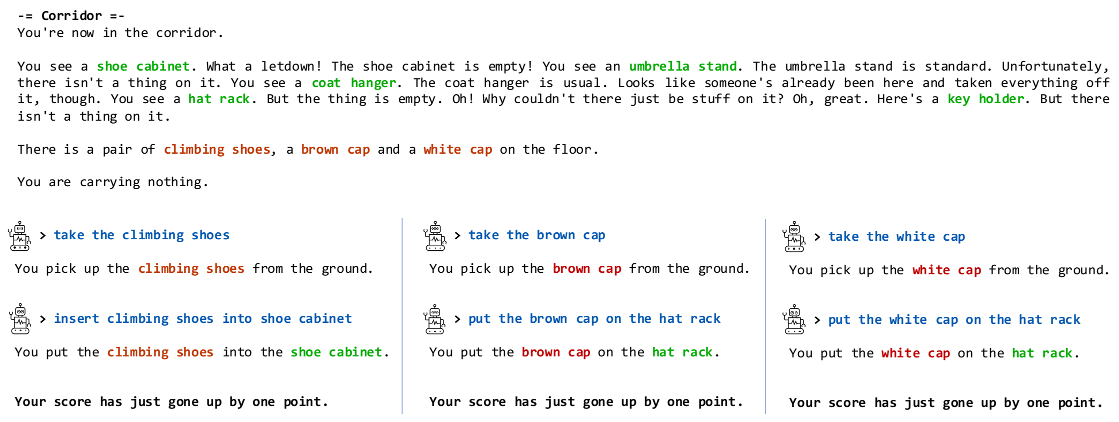
	Count	Examples
Rooms	8	*kitchen, bedroom*
Supporters/Containers	56	*dishwasher, wardrobe*
Unique Objects	190	*plate, dress*
Total Objects	872	*dirty plate, clean red dress*
Total Entities	928	*dirty plate, dishwasher*

Table 2.1: Number of entities, supporters/containers, and rooms in the TWC domain.

	#Objects	#Objects to find	#Rooms
Easy	1	1	1
Medium	2, 3	1, 2, 3	1
Hard	6, 7	5, 6, 7	1, 2

Table 2.2: Specification of the different difficulty levels in the TWC games. The number of objects that the agent has to find may be lower than the total number of objects in the environment, as the agent may be already carrying some objects at the beginning of the game.

the detailed specification of the games, whereas Figure 2.2 shows an example of a game belonging to the *medium* difficulty level. For each difficulty level, we provide a training set and two test sets. The training sets include games that were constructed out of $\frac{2}{3}$ of the unique objects reported in Table 2.1. The first test set was built based on the same pool of objects as in the training games. In the following, we will refer to this set as the *in*-distribution test set. The second test set uses the remaining $\frac{1}{3}$ objects and is called the *out*-of-distribution test set. This allows investigating the ability of the agents to achieve systematic generalization to unseen entities.


```
-= Corridor =-
You're now in the corridor.

You see a shoe cabinet. What a letdown! The shoe cabinet is empty! You see an umbrella stand. The umbrella stand is standard. Unfortunately,
there isn't a thing on it. You see a coat hanger. The coat hanger is usual. Looks like someone's already been here and taken everything off
it, though. You see a hat rack. But the thing is empty. Oh! Why couldn't there just be stuff on it? Oh, great. Here's a key holder. But there
isn't a thing on it.

There is a pair of climbing shoes, a brown cap and a white cap on the floor.

You are carrying nothing.
```

```
> take the climbing shoes

You pick up the climbing shoes from the ground.

> insert climbing shoes into shoe cabinet

You put the climbing shoes into the shoe cabinet.

Your score has just gone up by one point.
```

```
> take the brown cap

You pick up the brown cap from the ground.

> put the brown cap on the hat rack

You put the brown cap on the hat rack.

Your score has just gone up by one point.
```

```
> take the white cap

You pick up the white cap from the ground.

> put the white cap on the hat rack

You put the white cap on the hat rack.

Your score has just gone up by one point.
```

Figure 2.2: Sample game walkthrough for a game with *medium* difficulty level. Best viewed in colors. Highlights are not available to the agents and are shown for illustrative purpose only.

	Correctness	Completeness
Rated Commonsense	669	47
Rated NOT Commonsense	31	253

Table 2.3: Statistics from the human annotations to verify TWC

2.2.2 Verifying TWC

We assessed the TWC domain by means of a twofold evaluation aimed at verifying the quality of the dataset and benchmarking human performance on the proposed text-based environments.

The evaluation of the dataset had the purpose of ensuring that the proposed TWC domain in fact reflects commonsense knowledge. In particular, given $\langle \texttt{o}, \texttt{r}, \texttt{l} \rangle$ triples (where `o` denotes an object, `r` denotes a room, and `l` a location within that room, as defined in Section 2.2.1), we set up two annotation tasks aimed at verifying that:

- goal triples in the TWC dataset are commonsensical to humans (TWC is correct);
- non-goal triples, which are not in the dataset, are not commonsensical to humans (TWC is complete).

A total of 1000 triple annotations (700 on goal triples to assess correctness and 300 on non-goal triples to assess completeness) was collected from 10 annotators. Each triple was labeled by at least 3 annotators. Table 2.3 shows the overall agreement of the annotators with the TWC dataset, which demonstrates that TWC reflects human commonsense knowledge. For the overall annotation exercise, we can report inter-annotator agreement statistics, as the overall annotation is not imbalanced in terms of label marginals. We report a *Krippendorff's alpha* (Krippendorff 2018) value of $\alpha_\kappa = 0.74$, which shows that annotators had strong agreement when rating the triples.

2.2.3 Human performance on TWC

To complete our benchmarking of the TWC domain, we collected the performance of human players on the TWC text-based games. Human players were given access to a user interface aimed at recording all their interactions with the environment. At each step, the players were shown the current context of the game in text format and a drop-down list of all possible actions. A total of 16 annotators played games spread across the *easy*, *medium*, and *hard* levels. Each difficulty level had 5 games, both from the train and test distributions, for a total of 30 unique games. Each unique game was annotated by a minimum of 3 annotators. The results are presented in Table 2.4, along with the experimental results, to allow for direct comparison with the agents.

2.3 Infusing prior knowledge in neural agents

Text-based games can be seen as partially observable Markov decision processes (POMDP) (Kaelbling et al. 1998) where the system dynamics are determined by an MDP, but the agent cannot directly observe the underlying state. The agent receives a reward at every time step and its goal is to maximize the expected discounted sum of rewards. The TWC games allow the agent to perceive and interact with the environment via text. Thus, the observation at time step t, o_t, can be presented as a sequence of tokens, $o_t = (o_t^1, \dots, o_t^N)$. Similarly, denoting as $\mathcal{A}_t$ the set of available actions at time step t, each action $a \in \mathcal{A}_t$ is also a sequence of tokens $a = (a^1, \dots, a^M)$. The goal of this project is to design RL agents that can reason over a knowledge graph, hence we assume the agents also have access to a commonsense KG and are allowed to use it while selecting actions. To model TWC, we design a framework that can: (a) learn representations of various actions; (b) learn from sequential context; (c) dynamically retrieve the relevant commonsense knowledge; (d) integrate the retrieved commonsense knowledge with the context; and (e) predict next action. A block diagram of the framework is shown in Figure 2.3.

2.3.1 Observation encoder

We learn representations of the natural language observations by feeding them to a recurrent network. Given the current observation o_t, the models relies on pre-trained word embeddings to represent o_t as a sequence of d-dimensional vectors $\mathbf{x}_t^1, \dots, \mathbf{x}_t^N$, where each $\mathbf{x}_t^k \in \mathbb{R}^d$ is the word embedding of the k-th observed token o_t^k, $k = 1, \dots, N$. Then, a bidirectional GRU encoder (Cho et al. 2014) is used to process the sequence $\mathbf{x}_t^1, \dots, \mathbf{x}_t^N$ to get the representation of the current observation $\mathbf{o}_t = \mathbf{h}_t^N$, where $\mathbf{h}_t^k = \mathrm{GRU}(\mathbf{h}_t^{k-1}, \mathbf{x}_t^k)$, for $k = 1, \dots, N$, and $\mathbf{h}_t^0 = \mathbf{0}$.

2.3.2 Action encoder

Actions in our framework are expressed in natural language. We encode an action a_t^k in the set of admissible actions at time step t, $\mathcal{A}_t$ in a way similar to what described above for observations. Each toke of an action a_t^k is encoded using pre-trained word embeddings (Pennington et al. 2014) and we use a bidirectional GRU to get a final representation of an action a_t^k, which we denote as $\Game_t^k \in \mathbb{R}^d$.

2.3.3 Context encoder

A key challenge for our RL agent is in modeling context, i.e. the history of observations. The context is modeled using another recurrent encoder over the observations $\mathbf{o}_t$. A GRU network is used to encode the sequence of previous observations up to o_t into

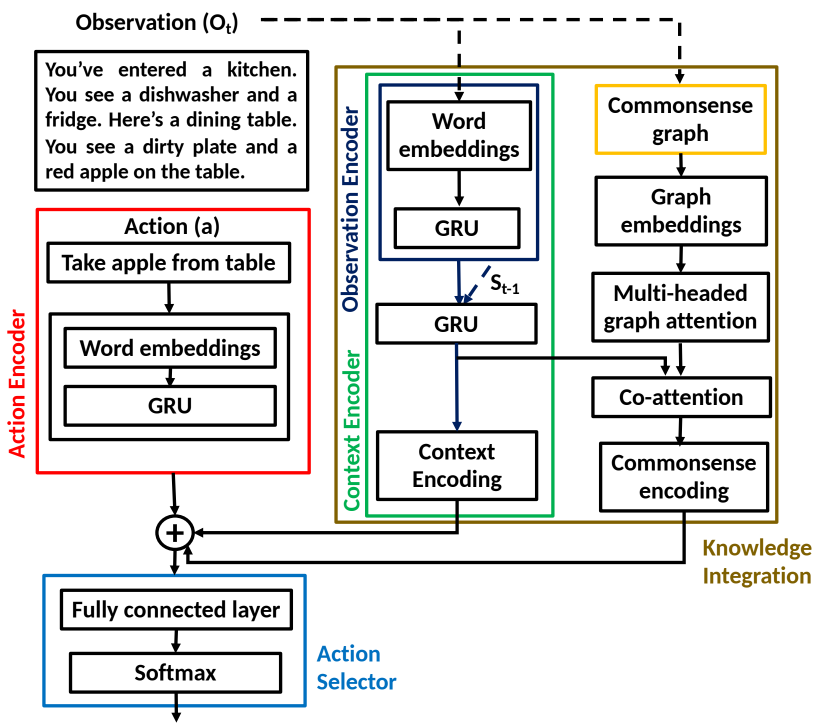


Figure 2.3: Overview of our framework's decision making at any given time step. The framework comprises of the following components (visually shown in color): (a) action encoder which encodes all admissible actions, (b) observation encoder which encodes the observation o_t, (c) context encoder, which encodes the dynamic context, (d) a dynamic commonsense subgraph of ConceptNet $\mathcal{G}_t^C$ extracted by the agent, (e) a knowledge integration component, which combines the information from textual observations and the extracted common sense subgraph, and (f) an action selection module. $\oplus$ denotes the concatenation operator.

a vector $\mathbf{s}_t = \text{GRU}(\mathbf{s}_{t-1}, \mathbf{o}_t)$, with $\mathbf{s}_0 = \mathbf{0}$. We refer to $\mathbf{s}_t$ as the state vector, or the context encoding. The context encoding will be used in addition to the commonsense knowledge in the final action prediction.

2.3.4 Dynamic commonsense subgraph

Our model retrieves commonsense knowledge from ConceptNet in the form of a graph $\mathcal{G}_t^C$. This graph is updated dynamically at each time step t. $\mathcal{G}_t^C$ is constructed by mapping the textual observation o_t at time t to ConceptNet and combining it with the graph at previous time step $\mathcal{G}_{t-1}^C$.

At each time step, we extract noun chunks from the observations o_t and then perform a max sub-string match with all the concepts in ConceptNet. This results in a set of entities for the observation o_t at time t. We then combine this set of entities with the ones already in $\mathcal{G}_{t-1}^C$ to get $\mathcal{V}_t^C$, the set of nodes of $\mathcal{G}_t^C$. $\mathcal{V}_t^C$ consists of all the concepts observed by the agent until time step t, including the description of the room, the current observation, and the objects in the inventory. Given $\mathcal{V}_t^C$, we describe three different techniques that automatically extract the commonsense graph $\mathcal{G}_t^C$ from external knowledge.

Direct Connections (DC). This is the baseline approach to construct $\mathcal{G}_t^C$. We fetch direct links between each of the concepts in $\mathcal{V}_t^C$ from ConceptNet.

Contextual Direct Connections (CDC). Since the goal of the agent is to clean up the house by putting objects into its appropriate locations, we hypothesize that adding links only between objects and containers may benefit the agent instead of links between all concepts as done by *Direct Connections*, as we might overwhelm the agent with noise. To accomplish this goal, we split the entities $\mathcal{V}_t^C$ into objects and containers. Since we know the entities from the inventory in $\mathcal{V}_t^C$ constitutes objects, no explicit labelling is needed as we consider the remaining entities as containers. We retain only the edges between objects and containers from ConceptNet.

Neighborhood (NG). Previous techniques only focus on connecting links between observed concepts $\mathcal{V}_t^C$ from external knowledge. In addition to the direct relations, it may be beneficial to include concepts from external knowledge that are related to $\mathcal{V}_t^C$, but have not been directly observed from the game. Therefore, for each concept in $\mathcal{V}_t^C$, we include all its neighboring concepts and associated links.

2.3.5 Knowledge integration

We enhance the text-based RL agent by allowing it to jointly contextualize information from both the commonsense subgraph and the observation representation. We call this step knowledge integration. We encode the commonsense graph using a graph encoder followed by a co-attention layer.

Graph encoder. The graph $\mathcal{G}_t^C$ is encoded as follows. First, pre-trained Numberbatch (Speer et al. 2017) embeddings are used to map the set of nodes $\mathcal{V}_t^C$ to a feature matrix $[\mathbf{v}_t^1, \ldots, \mathbf{v}_t^{|\mathcal{V}_t^C|}]^\top \in \mathbb{R}^{|\mathcal{V}_t^C| \times f}$. Here, $\mathbf{v}_t^i \in \mathbb{R}^f$ is the average embedding of words in node $v_i \in \mathcal{V}_t^C$. Then, these node embeddings are updated at each time step by message passing between the nodes of $\mathcal{G}_t^C$ with Graph Attention Networks (GATs) (Velickovic et al. 2018). This results in feature matrix $\mathbf{G} = [\mathbf{z}_t^1, \mathbf{z}_t^2, \cdots, \mathbf{z}_t^{|\mathcal{V}_t^C|}]^\top$ that better captures the conceptual relations between the nodes in the subgraph.

Context-graph attention. In order to combine the observational context and the retrieved commonsense graph, the agent relies on a bidirectional attention flow layer between these representations to re-contextualize the graph for the current state of the game (Yu et al. 2018). Following Yu et al. (2018), we compute a similarity matrix $\mathbf{S} \in \mathbb{R}^{N \times |\mathcal{V}_t^C|}$ between the context and entities in the extracted commonsense subgraph using a trilinear function (Yu et al. 2018). In particular, the similarity between the context encoding vector for each token $\mathbf{h}_t^i$ and a node encoding $\mathbf{z}_t^j$ in the commonsense subgraph is computed as $\mathbf{S}_{ij} = \mathbf{w}_S^\top[\mathbf{z}_t^j; \mathbf{h}_t^i; \mathbf{z}_t^j \circ \mathbf{h}_t^i]$ where $\circ$ denotes element-wise product and $\mathbf{w}_S$ is a learnable parameter. A $\mathrm{softmax}$ function is applied row-wise and column-wise on $\mathbf{S}$ to get respectively the scores $\bar{\mathbf{S}}_G$ for the context-to-graph attention and $\bar{\mathbf{S}}_O$ for the graph-to-context attention. Then, the attended context and graph vectors are computed as $\mathbf{A} = \bar{\mathbf{S}}_O^\top \cdot \mathbf{O}$ and $\mathbf{B} = \bar{\mathbf{S}}_O^\top \cdot (\bar{\mathbf{S}}_G \cdot \mathbf{G})$, where $\mathbf{G} = [\mathbf{z}_t^1, \mathbf{z}_t^2, \cdots, \mathbf{z}_t^{|\mathcal{V}^t|}]^\top$ and $\mathbf{O} = [\mathbf{h}_t^1, \mathbf{h}_t^2, \cdots, \mathbf{h}_t^N]^\top$ are the commonsense graph and observation encodings. The attention vectors are then combined together and the final graph encoding vectors $\mathbf{U}$ are calculated as $\mathbf{U} = \mathbf{W}_U^\top[\mathbf{G}; \mathbf{A}; \mathbf{G} \circ \mathbf{A}; \mathbf{G} \circ \mathbf{B}]$ where $\mathbf{W}_U$ is a learnable parameter.

2.3.6 Action selection

The agent applies a general attention over the nodes of $\mathcal{G}_t^C$ using the state vector and the action encoding $[\mathbf{s}_t; \mathbf{a}_i^t]$ (Luong et al. 2015). The attention score for each node is computed as $\alpha_i = [\mathbf{s}_t; \mathbf{a}_i^t]\mathbf{W}_g\mathbf{U}$, and the commonsense graph encoding for action $\mathbf{a}_i^t$ is given as $\mathbf{g}_i^t = \alpha_i^\top \mathbf{U}$. The action to take is chosen based on the context encoding $\mathbf{s}_t$, the commonsense graph encoding $\mathbf{g}_i^t$ and the action encoding $\mathbf{a}_i^t$. These encodings are concatenated into a single vector $\mathbf{r}_i^t = [\mathbf{s}_t; \mathbf{g}_i^t; \mathbf{a}_i^t]$. Then, a probability score is computed for each action $a_i \in \mathcal{A}_t$ using a feed-forward network as $\mathbf{p}_t = \mathrm{softmax}(\mathbf{W}_1 \cdot \mathrm{ReLU}(\mathbf{W}_2 \cdot \mathbf{r}_t + \mathbf{b}_2) + \mathbf{b}_1)$, where $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1$, and $\mathbf{b}_2$ are learnable parameters of the model. The final action is then given by the one with the maximum probability score, namely $\hat{a}_t = \arg\max_i p_{t,i}$.

2.4 Experiments

Given that the quality of the proposed TWC environment has already been evaluated in Section 2.2.2, the experiments discussed in this section primarily focus on verifying the following hypotheses:

- agents that utilize knowledge graphs can achieve better performance on TWC than their text-based counterparts;
- TWC can aid research in the use of knowledge graphs because of the gap between human performance and the best knowledge-aware agents.

The performance of the various agents is measured based both on the normalized score (reward) achieved by the agent and the number of steps (actions) taken. Each agent was trained for 100 episodes and the results were averaged over 10 runs. The maximum number of actions was limited to 50 for all agents. Following one of the winning strategies in the *FirstTextWorld Competition* (Adolphs et al. 2019), we use the Advantage Actor-Critic framework (Mnih et al. 2016) to train the agents.

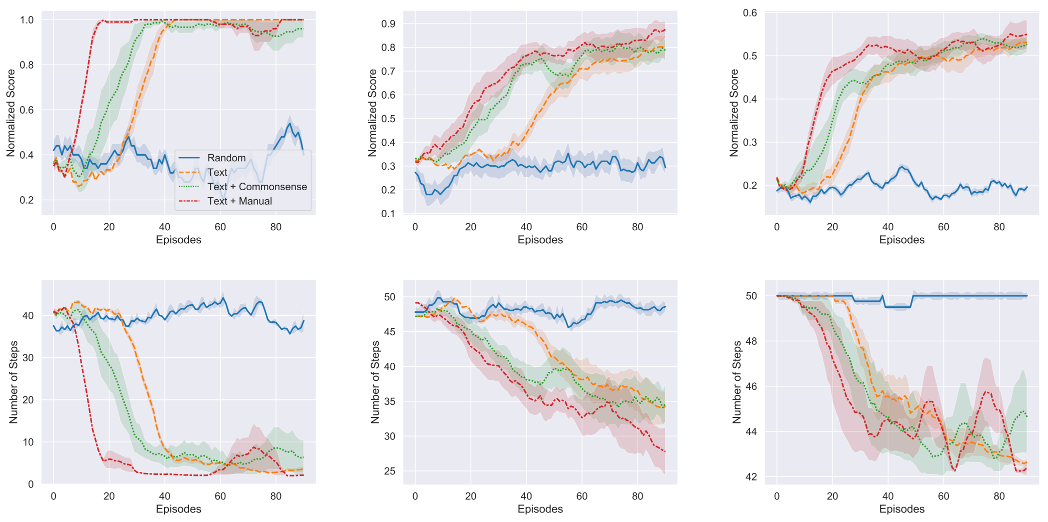


Figure 2.4: Performance evaluation (showing mean and standard deviation averaged over 10 runs) for the three difficulty levels: Easy (left), Medium (middle), Hard (right) using normalized score and the number of steps taken.

2.4.1 RL agents in TWC

We evaluate our framework on the TWC games. As a reference, we consider a random agent that randomly picks an action at each time step. We consider two types of experiment settings based on the type of information available to the RL agents: (1) `Text-based` RL agents have access to the textual description (observation) of the current state of the game provided by the TWC environment; and (2) `Commonsense-based` RL agents have access to both the observation and ConceptNet.

Text-only baseline agents. As baselines, we picked various SOTA text-based agents that utilize observation only: (1) **LM-NSP** uses language models such as *BERT* (Devlin et al. 2019) and *GPT2* (Radford et al. 2019) with the observation and the action pair as a Next Sentence Prediction (NSP) task; (2) **LSTM-A2C** (Narasimhan et al. 2015) uses the observed text to select the next action; (3) **DRRN** (J. He et al. 2016) utilizes the relevance between the observation and action spaces for better convergence; and (4) **KG-A2C** (Ammanabrolu and M. J. Hausknecht 2020) uses knowledge of the game environment generated from the observation to guide the agent's exploration. For these baselines, we use GloVe (Pennington et al. 2014) embeddings for text.

The results on these baselines are reported in Table 2.4. For each difficulty level, we report: the agents' performance; the optimal number of steps to solve the game[I]; and the human performance. The performance of GPT2-NSP and BERT-NSP shows that even powerful pretrained models if not tuned to this task have difficulty in these

[I]The optimal number of steps was computed by considering the objects already in the agent's possession, the number of objects to "put" (goals), and the number of rooms in the instance.

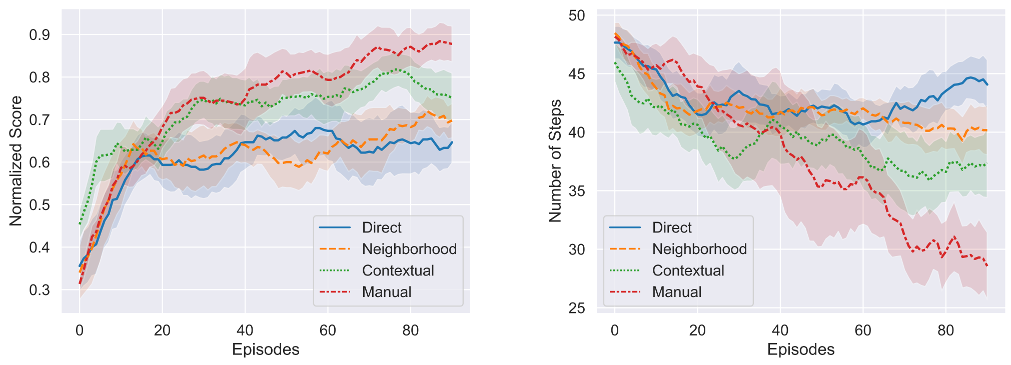


Figure 2.5: Training curves for the medium level games (showing mean and standard deviation averaged over 3 runs) with the different techniques for the commonsense sub-graph extraction.

commonsense RL games, as they do not capture commonsense relationships between entities. Baselines such as LSTM-A2C, DRRN, and KG-A2C have a competitive advantage over the LM-NSP baselines, as they effectively adapt to the sequential interaction with the environment to improve performance. Among these baselines, DRRN and KG-A2C perform better than LSTM-A2C as they utilize the structure of the state and action spaces for efficient exploration of the environment.

Commonsense-based agents. We introduce commonsense knowledge in two ways. The first is the agent Text + Numberbatch, which replaces GloVe embeddings in the LSTM-A2C agent with Numberbatch (*Nb*) embeddings (Speer et al. 2017) which were trained on text and ConceptNet. This is the naive approach to augment text information with commonsense knowledge. The results in Table 2.4 show that introducing *Nb* embeddings allows achieving a noticeable gain (an average of 3 steps in easy and 7 steps in medium level games) over GloVe embeddings.

In order to explicitly use commonsense knowledge, we experiment with the three different mechanisms outlined in Section 2.3.3 for retrieving relevant information from ConceptNet (`DC`, `CDC` and `NG`). These methods retrieve both the concepts and structure in the relevant sub-graphs from ConceptNet, which are leveraged by our co-attention mechanism (Section 2.3.5). The comparison of the agents' performance with different retrieving mechanisms is shown in Fig 2.5. The results show that `CDC` performs the best among other mechanisms, particularly compared to `DC`. Unlike `DC` that includes all the links between observed concepts from ConceptNet, `CDC` restricts links to those between observed objects and *containers*. This selection of relevant links from ConceptNet improves the performance of the agent.

Given that `CDC` performs best, we compare results on text-based models with `CDC`-augmented commonsense knowledge to other baselines. Table 2.4 shows results for text-based agents initialized with GloVe or *Nb* embeddings, and augmented with commonsense knowledge. We see that the commonsense-based RL agents perform better than text-based RL agents in the easy and medium level games. This is not surprising, as these instances mostly involve picking an object and placing it in a container in the same room. Both the text-based and commonsense RL agents struggle in the hard level, as these games have more than one room and multiple objects and containers. We also notice that the average number of steps taken by the commonsense-based RL agents are noticeably lower than the other agents as it efficiently uses commonsense knowledge to rule out implausible actions. This proves that TWC is a promising test-bed where commonsense knowledge helps.

Our results show that the agents still have much room for improvement in terms of retrieving and combining knowledge with observations. As a starting point for showing that there is headroom, we switched the retrieval mechanism to manually selected information from ConceptNet. We manually retrieved the relevant commonsense knowledge by extracting the commonsensical paths between entities in ConceptNet, corresponding to objects in the TWC games and their goal locations. The manual subgraph includes all the relevant shortest paths between an object and its location, within a 2-hop neighborhood expansion of both nodes. Since the extracted subgraph can be very large even for the easy games, further pruning was performed to remove noise. We emphasize that the manual annotation can be error-prone or result in manual subgraphs that lack potentially useful information. Thus, the manual graphs should not be taken as a gold standard. In Table 2.4, agents that are augmented with the manual graph perform better than the other automated retrieval mechanisms (average reduction of $2-5$ steps on easy and medium). Figure 2.4 shows training curves for the Text-only, Text+Commonsense and Text+Manual agents on the three difficulty levels. We notice that infusing commonsense knowledge allows achieving faster convergence both in terms of the number of steps taken by the agents and the final score.

Human performance on TWC. We also present the results of human performance in TWC. The **O** and **H** columns in Table 2.4 (two per condition) present these results. A quick comparison of these numbers reveals two major results: (1) human performance **H** is very close to the optimal number of steps **O** in all 3 conditions; and (2) there is significant headroom between **H** and all of the other agents in the table, including the ones with the manual graph. This confirms that there is still much progress to be made in retrieving and encoding the commonsense knowledge effectively to solve such problems; and that TWC can spur further research.

2.4.2 Generalization

Table 2.4 reports the results both for test games that belong to the same distribution used at training time (**IN**), and games that were generated from a different set of entities

		Easy				Medium				Hard			
		O	**H**	**#Steps**	**Norm. Score**	**O**	**H**	**#Steps**	**Norm. Score**	**O**	**H**	**#Steps**	**Norm. Score**
IN	**GPT2-NSP**	2.00 ± 0.00	2.12 ± 0.49	30.36 ± 0.00	0.64 ± 0.00	3.60 ± 0.55	5.33 ± 2.06	42.12 ± 0.00	0.70 ± 0.00	15.00 ± 2.00	15.00 ± 3.29	50.00 ± 0.00	0.36 ± 0.00
	BERT-NSP			25.20 ± 0.00	0.76 ± 0.00			34.72 ± 0.00	0.88 ± 0.00			50.00 ± 0.00	0.52 ± 0.00
	LSTM-A2C			17.59 ± 3.11	0.86 ± 0.04			37.99 ± 6.03	0.74 ± 0.11			49.21 ± 0.58	0.54 ± 0.04
	DRRN			18.88 ± 2.69	0.81 ± 0.08			33.41 ± 2.81	0.73 ± 0.06			46.20 ± 4.86	0.44 ± 0.01
	KG-A2C			17.65 ± 3.62	0.85 ± 0.07			37.18 ± 4.86	0.72 ± 0.07			49.36 ± 7.50	0.46 ± 0.10
	Text												
	+*Commonsense*			14.18 ± 6.47	0.89 ± 0.10			34.67 ± 6.65	0.78 ± 0.07			48.45 ± 2.50	0.51 ± 0.10
	+*Manual*			13.70 ± 1.85	0.92 ± 0.03			29.26 ± 0.94	0.88 ± 0.03			46.43 ± 3.67	0.54 ± 0.04
	+*Numberbatch*			11.79 ± 3.04	0.96 ± 0.03			27.10 ± 5.06	0.85 ± 0.06			44.22 ± 4.86	0.57 ± 0.00
	+*Nb+Commonsense*			14.43 ± 3.08	0.93 ± 0.06			25.11 ± 2.33	0.87 ± 0.04			43.27 ± 0.70	0.45 ± 0.00
	+*Nb+Manual*			13.37 ± 5.63	0.92 ± 0.07			23.51 ± 1.28	0.91 ± 0.06			42.87 ± 0.65	0.52 ± 0.01
OUT	**GPT2-NSP**	2.00 ± 0.00	2.24 ± 0.75	40.28 ± 0.00	0.46 ± 0.00	4.40 ± 1.14	4.40 ± 1.85	44.96 ± 0.00	0.38 ± 0.00	14.60 ± 2.67	17.67 ± 3.31	50.00 ± 0.00	0.14 ± 0.00
	BERT-NSP			24.76 ± 0.00	0.72 ± 0.00			41.12 ± 0.00	0.55 ± 0.00			50.00 ± 0.00	0.27 ± 0.00
	LSTM-A2C			19.89 ± 1.86	0.79 ± 0.01			43.70 ± 5.52	0.52 ± 0.18			50.00 ± 0.00	0.27 ± 0.01
	DRRN			19.49 ± 4.89	0.84 ± 0.08			40.49 ± 4.41	0.56 ± 0.07			50.00 ± 0.00	0.18 ± 0.10
	KG-A2C			18.00 ± 3.24	0.87 ± 0.05			43.08 ± 4.13	0.54 ± 0.17			49.96 ± 0.00	0.22 ± 0.00
	Text												
	+*Commonsense*			19.14 ± 3.32	0.83 ± 0.07			41.01 ± 6.97	0.56 ± 0.13			49.99 ± 0.01	0.28 ± 0.05
	+*Manual*			16.86 ± 2.26	0.89 ± 0.04			39.95 ± 2.46	0.71 ± 0.06			49.97 ± 0.04	0.26 ± 0.11
	+*Numberbatch*			19.77 ± 2.50	0.81 ± 0.15			34.54 ± 2.89	0.80 ± 0.04			49.95 ± 0.08	0.29 ± 0.02
	+*Nb+Commonsense*			20.84 ± 1.13	0.83 ± 0.03			33.43 ± 2.11	0.71 ± 0.09			50.00 ± 0.00	0.25 ± 0.01
	+*Nb+Manual*			18.24 ± 4.63	0.83 ± 0.09			30.12 ± 4.62	0.84 ± 0.03			49.99 ± 0.02	0.22 ± 0.05

Table 2.4: Generalization results for within distribution (*IN*) and out-of-distribution (*OUT*) games. *O* represents the optimal #steps needed to accomplish the goals. *H* represents human-level performance. All agents were restricted to a max of 50 steps.

(**OUT**). We see a similar trend on both these settings. The commonsense-enhanced agent outperforms the text-only agent in all cases. However, all agents including those that utilize commonsense knowledge show similar drop in performance from **IN** to **OUT** distribution. This is in contrast to the use of the knowledge graphs in other NLP tasks such as textual entailment where knowledge graphs have shown to be robust to changes in the underlying (training and testing) environment (Kapanipathi et al. 2020; Q. Chen et al. 2018). The task of designing knowledge-enabled agents that are robust to such changes is another open challenge for the community that can be evaluated by TWC. We will make some steps to address this challenge in Chapter 4.

Results summary. Our results establish that TWC is an environment where agents augmented with commonsense knowledge show better performance than their text-based counterparts. Based on the experiments with manually retrieved sub-graphs, optimal steps, and the human performance numbers, we show that TWC has enough headroom for future research efforts to: (1) retrieve more relevant commonsense knowledge for KBs; and (2) for new agents/techniques to exploit such knowledge.

2.5 Related work

Textual reinforcement learning. Games are a rich domain for studying grounded language and how information from text can be utilized in control. Recent work has explored text-based RL games to learn strategies for *CivII* (Branavan et al. 2012), multi-user dungeon games (Narasimhan et al. 2015), etc. Our work builds on the

TextWorld (Côté et al. 2018) sandbox learning environment. Since its introduction, there has been a large body of work devoted to improving performance on this benchmark. A recent line of work on TextWorld learns symbolic representations of the agent's belief. Notably, Ammanabrolu and Riedl (2019) proposed *KG-DQN* and Adhikari et al. (2020) proposed *GATA*. Both approaches represent the game state as a belief graph. This graph is used to prune the action space, enabling efficient exploration, in a different way from our work which uses common sense. The *LeDeepChef* system (Adolphs et al. 2019) is also related to our work. They achieve transfer by additionally supervising the model with a list of the most common food items in FreeBase (Bollacker et al. 2008), allowing their agent to generalize to hitherto unseen recipes and ingredients. Zahavy et al. (2018) propose the Action-Elimination Deep Q-Network (AE-DQN), which learns to predict invalid actions in the text-adventure game *Zork*. This network allows the model to efficiently handle the large action space. The use of commonsense knowledge in our work potentially has the same effect of down-weighting implausible actions.

External knowledge for efficient RL. There have been few attempts on adding prior or external knowledge to RL approaches. Notably, Garnelo et al. (2016) proposed *Deep Symbolic RL*, which combines aspects of symbolic AI with neural networks and RL as a way to introduce commonsense priors. There has also been work on *policy transfer* (Bianchi, Celiberto Jr, et al. 2015), which studies how knowledge acquired in one environment can be re-used in another one; and *experience replay* (Z. Wang et al. 2017; L.-J. Lin 1992; L.-J. Lin 1993) which studies how an agent's previous experiences can be stored and then later reused. In this chapter, we use commonsense knowledge as a way to improve sample efficiency in text-based RL agents. To the best of our knowledge, there is no prior work that *practically* explores how commonsense can be used to make RL agents more efficient. The most relevant prior work is by Martin et al. (2018), who use commonsense rules to build agents that can play tabletop role-playing games. However, the commonsense rules in their work are manually engineered, while our commonsense knowledge is extracted from ConceptNet.

Commonsense in NLP. Recently, there has been a lot of work in NLP to utilize commonsense for QA, NLI, etc. (Sap et al. 2019; Talmor et al. 2018). Many of these approaches seek to effectively utilize ConceptNet by reducing the noise retrieved from it (B. Y. Lin et al. 2019; Kapanipathi et al. 2020). This is also a key challenge in TWC.

2.6 Conclusion

We created a novel environment (TWC) to evaluate the performance on RL agents on text-based games requiring commonsense knowledge. We introduced a framework of agents which tracks the state of the world; uses the sequential context to dynamically retrieve relevant commonsense knowledge from a knowledge graph; and learns to combine the two different modalities. Our agents equipped with common sense achieve their goals with greater efficiency and less exploration when compared to a text-only model, thus showing the value of our new environments and models. Therefore, we believe that our TWC environment provides interesting challenges and can be effectively used to fuel further research in this area.

3 Joint reasoning over prior and belief knowledge graphs

3.1 Introduction

In Chapter 2, we introduced a novel RL environment for benchmarking commonsense reasoning abilities of neural agents. The hypothesis behind TWC is that prior commonsense knowledge allows the agent to understand how current actions might affect future world states. This enables improved look-ahead planning (Juba 2016), thus leading to sample-efficient selection of actions at each step and driving the agent closer to optimal performance.

A recent line of work in TBGs aims has shown that reasoning over symbolic representations of the agent's belief can greatly improve generalization performance. Notably, Ammanabrolu and Riedl (2019) proposed *KG-DQN* and Ammanabrolu and M. J. Hausknecht (2020) proposed *KG-A2C*. The idea behind both approaches is to represent the game state as a belief graph, which is constructed from natural language observations using information extraction tools and hand-crafted rules. Recently, Adhikari et al. (2020) proposed the graph-aided transformer agent (*GATA*), an approach to construct and update a latent belief graph in the form of a learned adjacency matrix. This graph is not constructed from scratch at each step, but rather it is updated after each interaction with the environment based on the current observation. The graph updater is pre-trained and not fine-tuned during exploration. Similar approaches for building dynamic belief graphs have also been explored in the context of machine comprehension of procedural text (Das, Munkhdalai, et al. 2018).

In this chapter, we posit that to efficiently act in such text-based gaming environments, an agent must be able to effectively track the state of the game and use it to jointly retrieve and leverage the relevant commonsense knowledge. Figure 3.1 shows how state and prior knowledge are complementary and can be used jointly to improve sample efficiently and generalization. Thus, we propose a technique to: (a) track the state of the game in the form of a symbolic graph that represents the agent's current

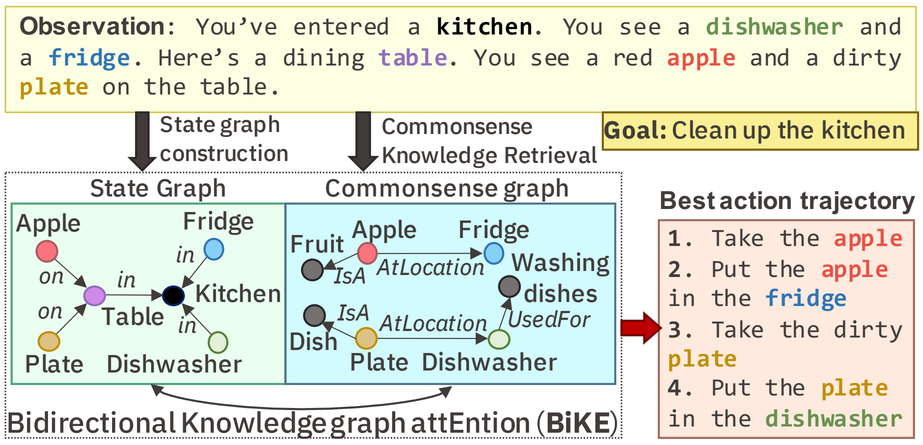

Observation: You've entered a kitchen. You see a dishwasher and a fridge. Here's a dining table. You see a red apple and a dirty plate on the table.
State graph construction
Commonsense Knowledge Retrieval
Goal: Clean up the kitchen
State Graph
Apple
Fridge
on
in
in
Kitchen
on
Table
in
Plate
Dishwasher
Commonsense graph
Fruit
Apple
Fridge
IsA
AtLocation
Washing dishes
IsA
AtLocation
UsedFor
Dish
Plate
Dishwasher
Bidirectional Knowledge graph attEntion (BiKE)
Best action trajectory
1. Take the apple
2. Put the apple in the fridge
3. Take the dirty plate
4. Put the plate in the dishwasher

Figure 3.1: An illustration of a TBG that requires both the state representation of the game as well as the external commonsense knowledge for efficient exploration and learning the best action trajectory. The observation text feeds into the state and commonsense graphs; and the best action trajectory is computed based on information from both graphs.

belief of the state of the world (Ammanabrolu and M. J. Hausknecht 2020; Adhikari et al. 2020); (b) retrieve the relevant commonsense knowledge from ConceptNet (Speer et al. 2017), and (c) jointly leverage the state graph and the retrieved commonsense graph. This combined information is then used to select the optimal action. In this chapter, we show that the agent that models the state graph and use commonsense graph into the reinforcement learning framework achieves better sample complexity and highest rewards across three difficulty levels. Qualitative analyses reveal how the agent balances the information from state and commonsense graphs to efficiently explore this challenging environment.

3.2 Model and architecture

Text-based games can be framed as partially observable Markov decision processes (POMDPs) (Spaan 2012). A POMDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{E}, r)$, where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ denotes the action space, $\mathcal{O}$ denotes the observation space, $\mathcal{T}$ denotes the state transition probabilities, $\mathcal{E}$ denotes the conditional observation emission probability, and $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function. The observation o_t at time step t depends on the current state. Both observations and actions are rendered in text. The agent receives a reward at every time step t: $r_t = r(o_t, a_t)$, and the agent's goal is to maximize the expected discounted sum of rewards: $\mathbb{E}[\sum_t \gamma^t r_t]$, where $\gamma \in [0, 1]$ is a discount factor.

The high-level architecture of our model contains three major components: (a) the input encoder; (b) a graph-based knowledge extractor; and (c) the action prediction module. The input encoding layers are used to encode the observation o_t at time step t

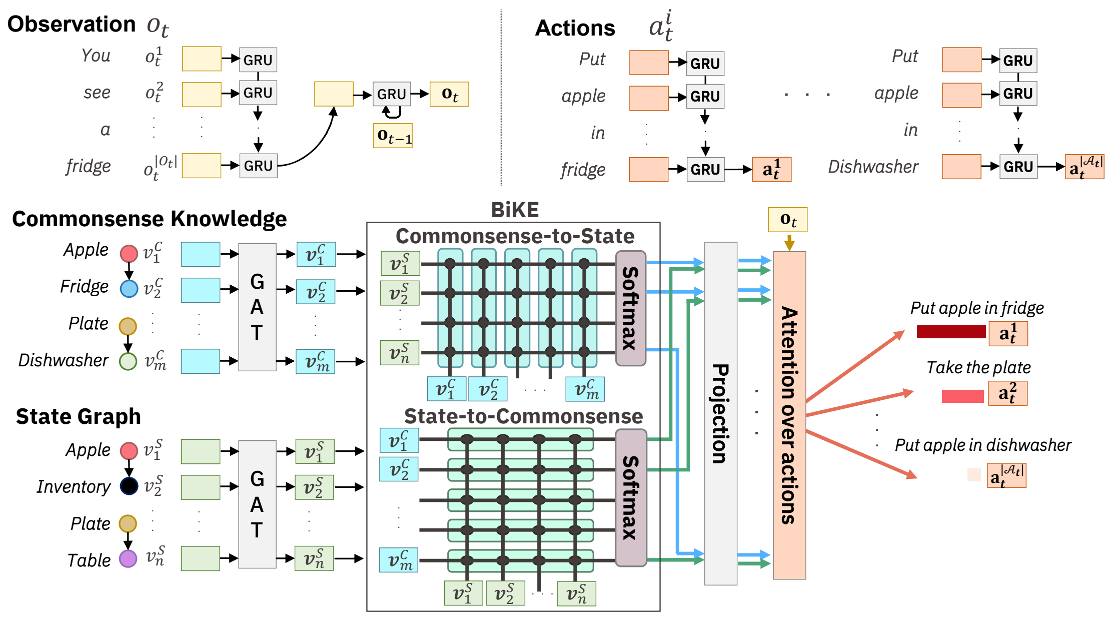


Figure 3.2: Visualization of our overall approach with BiKE

and the list of admissible actions using GRUs (Ammanabrolu and M. J. Hausknecht 2020), as described in Chapter 2. This allows us to obtain a representation $\boldsymbol{o}_t$ for the observation o_t at time step t. Similarly, we obtain a representation $\boldsymbol{a}_t^k$ for each action a_t^k in the set of admissible actions at time step t, $\mathcal{A}_t$.

The graph-based knowledge extractor collects relevant knowledge from complementary knowledge sources: the game state and external commonsense knowledge. We allow information from each knowledge source to guide and direct better representation learning for the other. Recent efforts have demonstrated the use of primarily two different types of knowledge sources for TextWorld RL Agents. A **State Graph** (SG) captures state information (Ammanabrolu and Riedl 2019) about the environment represented via a semantic graph. The example in Figure 3.2 shows that information such as `Apple` $\rightarrow$ `on` $\rightarrow$ `Table` is extracted from the textual observations from the environment. Specifically, Ammanabrolu and Riedl (2019) create such knowledge graphs by extracting information using OpenIE (Angeli et al. 2015) and some manual heuristics. A **Commonsense Graph** (CG) captures external commonsense knowledge (Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021) between entities (from commonsense knowledge sources such as ConceptNet). We posit that RL agents can make use of information from both these graphs during different sub-tasks, enabling efficient learning. The state graph provides the agent with a symbolic way of representing its current perception of the game state, including its understanding of the surroundings. On the other hand, the commonsense graph provides the agent with complementary knowledge about what actions make sense in a given state, thus enabling more efficient exploration of the very large natural language action space.

We combine the state information with commonsense knowledge using a **Bi**directional **K**nowledge-graph att**E**ntion (**BiKE**) mechanism, which re-contextualizes the *state* and *commonsense* graphs based on each other for optimal action trajectories. Figure 3.2 provides a compact visualization.

3.3 Knowledge integration using BiKE

The graph-based knowledge extractor produces m entities $v_1^C, v_2^C, \cdots, v_m^C$ for the commonsense graph $\mathcal{G}_t^C$ and n entities $v_1^S, v_2^S, \cdots, v_n^S$ for the state graph $\mathcal{G}_t^S$ (for ease of notation, we omit the subscript t from the nodes of both graphs). Note that the entities extracted for $\mathcal{G}_t^C$ are based on the vocabulary used in ConceptNet, and may not necessarily be the same set of entities in $\mathcal{G}_t^S$ (see Figure 3.1). We embed the extracted entities in both graphs using *Numberbatch* (Speer et al. 2017). We then encode both graphs using a graph attention network (Velickovic et al. 2018), in order to share information among the nodes by message passing. We thus get a representation $\boldsymbol{v}_i^S, i = 1, \ldots n$, for each node in $\mathcal{G}_t^S$, and a representation $\boldsymbol{v}_j^C, j = 1, \ldots m$, for each node in $\mathcal{G}_t^C$.

We aim to use the state and the commonsense knowledge graphs to improve the agent's exploration strategy. Inspired by bidirectional attention mechanism in question answering (Seo et al. 2017), we introduce a bidirectional attention flow between $\mathcal{G}_t^S$ and $\mathcal{G}_t^C$ to fuse the knowledge from these two graphs. The information flow across the graphs allows the model to learn commonsense-aware state graph representations, and state-aware commonsense knowledge graph representations.

In details, we compute a graph similarity matrix $\boldsymbol{S} \in \mathbb{R}^{n \times m}$ across the graph entities to learn a state-to-commonsense graph attention function and a commonsense-to-state graph attention function. An entry of the matrix $\boldsymbol{S}_{ij} = g(\boldsymbol{v}_i^S, \boldsymbol{v}_j^C)$ captures how each node v_i^S in the graph $\mathcal{G}_t^S$ is linked to a node v_j^C in the other graph $\mathcal{G}_t^C$, and vice versa. Here g is a learnable function that maps $\boldsymbol{v}_i^S$ and $\boldsymbol{v}_j^C$ to a similarity score. In our implementation, we learn g using a simple feed-foward network with 2 layers and ReLU activations. We compute the state-to-commonsense graph attention values $\boldsymbol{A}$ by taking a softmax along the rows of $\boldsymbol{S}$: this signifies the attention bestowed by each state graph node on the nodes of the commonsense graph. Similarly, we get the commonsense-to-state graph attention values $\bar{\boldsymbol{A}}$ by taking a softmax along the columns of $\boldsymbol{S}$. Then, we calculate an aggregated representation of the whole state graph $\mathcal{G}_t^S$ as:

$$\boldsymbol{g}_t^S = \frac{1}{n} \sum_{i=1}^{n} f_S\Big(\boldsymbol{v}_i^S, \sum_{j=1}^{m} \boldsymbol{A}_{ij} \boldsymbol{v}_j^C\Big),$$

where f_S is a learnable function implemented as a 2-layer feed-forward network with ReLU activation functions. Similarly, we obtain an embedding for the commonsense graph $\mathcal{G}_t^C$ as:

$$\boldsymbol{g}_t^C = \frac{1}{m} \sum_{j=1}^{m} f_C\Big(\boldsymbol{v}_j^C, \sum_{i=1}^{n} \bar{\boldsymbol{A}}_{ij} \boldsymbol{v}_i^S\Big).$$

where f_C is another learnable function implemented as a 2-layer feed-forward network with ReLU activation functions.

Finally, we compute a score for an action $a_t^k \in \mathcal{A}_t$ as $h(\boldsymbol{o}_t, \boldsymbol{a}_t^k, \boldsymbol{g}_t^S, \boldsymbol{g}_t^C)$, where h is a learnable function that projects the concatenation $[\boldsymbol{o}_t; \boldsymbol{a}_t^k; \boldsymbol{g}_t^S; \boldsymbol{g}_t^C]$ to the score for action $a_t^k \in \mathcal{A}_t$.

3.4 Experiments

We generate a set of games with 3 difficulty levels using the TWC (Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021) framework: (i) *easy* level, which has 1 room containing 1 to 3 objects; (ii) *medium* level, which has 1 or 2 rooms with 4 or 5 objects; and (iii) *hard* level, a mix of games with a high number of objects (6 or 7 objects in 1 or 2 rooms) or high number of rooms (3 or 4 rooms containing 4 or 5 objects).

We compare 5 text-based RL agents: (a) a text-only agent (**Text**), which selects the best action based only on the encoding of the history of observations; (b) **DRRN** (J. He et al. 2016; Narasimhan et al. 2015), which relies on the relevance between the observation and action spaces; (c) an agent enhanced with access to an external commonsense knowledge graph (**+Commonsense**) (Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021); (d) an agent that, following Ammanabrolu and M. J. Hausknecht (2020), models the state of the world as a symbolic graph (**+State**); and (e) the agent **BiKE** described in Section 3.2, which relies on both state and commonsense graph representations. The agents are trained over 100 episodes with a 50-step maximum. All policies are learned using Actor-Critic (Mnih et al. 2016).

3.4.1 Sample efficiency and generalization performance

Figure 3.3 shows the learning curves for the text-only agent and the agents equipped with state and/or commonsense graph representations at training time. For reference, we also report the performance of an agent that selects a random action at each time step (**Random**). We notice that, overall, agents equipped with either state or commonsense graph representations perform better than their text-only counterparts, both in terms of the number of steps taken and the normalized score. In particular, the BiKE agent outperforms all other agents in all difficulty levels, showing that symbolic state representations and prior commonsense knowledge can be jointly used for better sample efficiency and results. Table 3.1 shows the performance of the agents on the test set. Following Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. (2021),

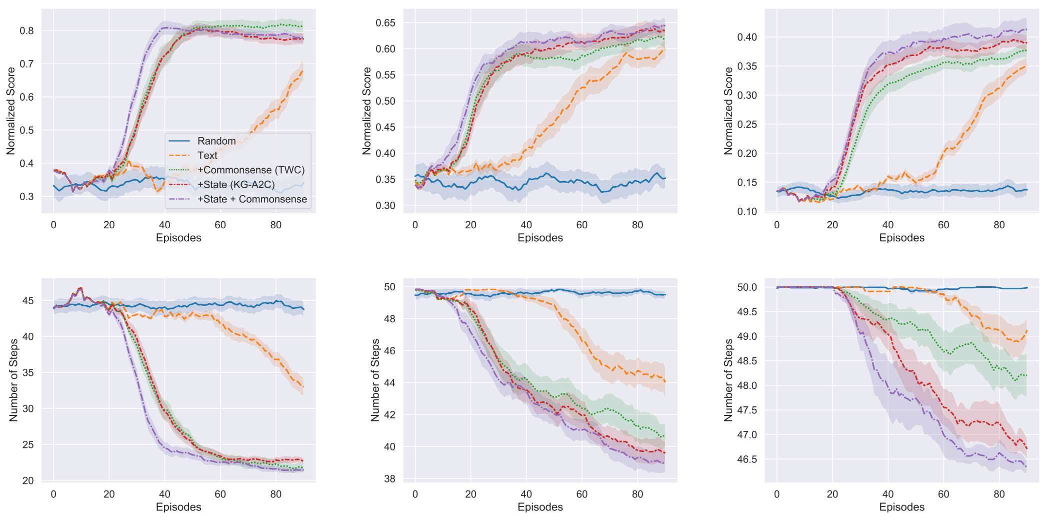


Figure 3.3: Performance evaluation (showing mean and standard deviation averaged over 3 runs) for the three difficulty levels: Easy (left), Medium (middle), Hard (right) using normalized score and the number of steps taken.

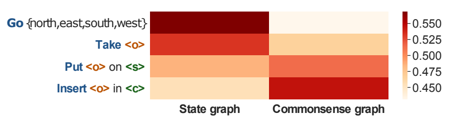
		Easy		Medium		Hard	
		#Steps	Norm. Score	#Steps	Norm. Score	#Steps	Norm. Score
IN	Text	23.83 ± 2.16	0.88 ± 0.04	44.08 ± 0.93	0.60 ± 0.02	49.84 ± 0.38	0.30 ± 0.02
	DRRN	22.08 ± 4.17	0.82 ± 0.06	44.04 ± 1.64	0.59 ± 0.02	49.82 ± 0.61	0.29 ± 0.01
	+Commonsense (TWC)	20.59 ± 5.01	0.89 ± 0.06	42.61 ± 0.65	0.62 ± 0.03	48.45 ± 1.13	0.32 ± 0.04
	+State (KG-A2C)	22.10 ± 2.91	0.86 ± 0.06	41.61 ± 0.37	0.62 ± 0.03	48.00 ± 0.61	0.32 ± 0.00
	+State + Commonsense (BiKE)	18.27 ± 1.13	0.94 ± 0.02	39.34 ± 0.72	0.64 ± 0.02	47.19 ± 0.64	0.34 ± 0.02
OUT	Text	29.90 ± 2.92	0.78 ± 0.02	45.90 ± 0.22	0.55 ± 0.01	50.00 ± 0.00	0.20 ± 0.02
	DRRN	29.71 ± 1.81	0.76 ± 0.05	45.18 ± 1.19	0.56 ± 0.02	50.00 ± 0.00	0.21 ± 0.02
	+Commonsense (TWC)	27.74 ± 4.46	0.78 ± 0.07	44.89 ± 1.52	0.58 ± 0.01	50.00 ± 0.00	0.19 ± 0.03
	+State (KG-A2C)	28.34 ± 3.63	0.80 ± 0.07	43.05 ± 2.52	0.59 ± 0.01	50.00 ± 0.00	0.21 ± 0.00
	+State + Commonsense (BiKE)	25.59 ± 1.92	0.83 ± 0.01	41.01 ± 1.61	0.61 ± 0.01	50.00 ± 0.00	0.23 ± 0.02

Table 3.1: Test-set performance results for within distribution (*IN*) and out-of-distribution (*OUT*) games.

we compared our agents on two test sets: (**IN**) uses the same entities as the training set, and (**OUT**) uses entities that were not included in the training set. The experimental results show that the BiKE agent generalizes better than all the baselines across the 3 difficulty levels.

3.4.2 Qualitative analysis

From Figure 3.3 and Table 3.1, we notice that the **+Commonsense** agent performs better on the *easy* level, whereas the **+State** agent performs better on the *medium* and *hard* levels. This suggests that the state representation can be leveraged to drive exploration and interaction with objects in environments with multiple rooms; whereas prior commonsense knowledge allows the agent to act more efficiently by selecting the appropriate commonsensical locations of different objects. In order to investigate this



(a) Average relevance of the main action templates to the state and commonsense graphs across the *hard* games.

Timestep	t	$\longrightarrow$ $t+1$	$\longrightarrow$ $t+2$
Room	*Living Room*	$\longrightarrow$ *Living Room*	$\longrightarrow$ *Bedroom*
Action Taken	**take checkered jumper**	$\longrightarrow$ **go** `west`	$\longrightarrow$ **insert checkered jumper** `into` **wardrobe**
Most relevant graph	*State graph*	*State graph*	*Commonsense graph*
Most relevant nodes	**checkered jumper**	**checkered jumper, exit to west**	**wardrobe**

(b) Example of most relevant graphs and nodes (by action taken) for one example game excerpted from the *hard* difficulty level.

Figure 3.4: Relevance given to the: (a) state and commonsense graphs; and to (b) their nodes (by action taken).

hypothesis, we computed the average importance given by the agent to the state graph and the commonsense graph when selecting the different action templates shown in Figure 3.4a. For each action template, the figure shows the normalized attention weight given to the two graphs, averaged across 5 runs of all games in the *hard* difficulty level. Actions requiring information about the goal of the game, like `put` and `insert`, benefit more from attending to the commonsense graph; whereas actions aimed at exploring the environment and collecting objects, like `go` and `take`, benefit more from the state representation.

As further qualitative analysis, we report an example of the most attended nodes and graphs from an excerpt of a game belonging to the *hard* difficulty level in Figure 3.4b. As noted above, the `take` and `go` actions rely more on the state graph, whereas the `insert` action relies on the commonsense graph. Among the nodes in these graphs, the entities that are finally mentioned in the action receive the highest attention score. This shows how our agent is able to transfer the bidirectional attention over graphs into specific game instances.

3.5 Conclusion

In this chapter, we showed that in order to be sample-efficient in TBGs, agents must be able to jointly track the state of the game and relevant commonsense knowledge. We proposed a technique that models both forms of knowledge as graphs, and combines them using Bidirectional Knowledge-graph attEntion (BiKE). The resulting agent was found to be more sample-efficient than approaches that considered neither or only one of these graphs.

4 Case-based reasoning for textual reinforcement learning

4.1 Introduction

As we have seen in Chapters 2 and 3, state-of-the-art agents in textual reinforcement learning are still very inefficient and suffer from insufficient generalization to novel environments (Ammanabrolu and M. J. Hausknecht 2020; Adhikari et al. 2020; Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021; K. Murugesan et al. 2021). Even the agent of Chapter 3 requires 5 to 10 times more steps than a human to accomplish even simple household tasks (Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021). As the agents are purely neural architectures requiring significant training experience and computation, they fail to efficiently adapt to new environments and use their past experiences to reason in novel situations. This is in stark contrast to human learning which is much more robust, efficient and generalizable (Lake, Ullman, et al. 2017).

Motivated by this fundamental difference in learning, we propose new agents for TBGs that rely on case-based reasoning (CBR) (Aamodt et al. 1994) to efficiently act in the world. CBR draws its foundations in cognitive science (Schank 1983; Kolodner 1983) and mimics the process of solving new tasks based on solutions to previously encountered similar tasks. Concretely, we design a general CBR framework that enables an agent to collect instances of past situations which led to a positive reward (known as cases) while interacting with the world. During decision making, the agent retrieves a case most similar to the current situation and then applies it after appropriately mapping the case to the current context.

The CBR agent stores past experiences, along with the actions it performed, in a repository called its memory. In order to efficiently use these stored experiences, the agent should be able to represent relevant contextual information from the state of the game in a compact way, while retaining the property that contexts that require similar actions receive similar representations. We represent the state of the game as a knowledge graph (Ammanabrolu and M. J. Hausknecht 2020) and we address these challenges by utilizing *(a)* seeded message propagation that focuses only on a subset of relevant nodes and *(b)* vector quantization (Ballard 2000) to efficiently map similar contexts to similar discrete representations. Vector quantization allows the model to significantly compress the context representations while retaining their semantics; thereby, allowing for a scalable implementation of CBR in RL settings. An illustration of the proposed framework is shown in Figure 4.1.

Our experiments show that CBR can be used to consistently boost the performance of various on-policy RL agents proposed in the literature for TBGs. We obtain a new state-of-the-art on the *TextWorld Commonsense* (Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021) dataset and we achieve better or comparable scores on 24 of the 33 games in the *Jericho* suite (M. Hausknecht et al. 2020) compared to previous work. We also show that CBR agents are resilient to domain shifts and suffer only marginal drops in performance **(6%)** on out-of-distribution settings when compared to their counterparts **(35%)**.

4.2 Preliminaries

Problem statement. Following the formulation of Chapters 2 and 3, we model a TBG as a Partially Observable Markov Decision Process (POMDP) $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{E}, r)$, where $\mathcal{S}$ is the set of states of the environment of the game, $\mathcal{A}$ is the natural language action space, $\mathcal{O}$ is the set of observations or sequences of words describing the current state, $\mathcal{T}$ are the conditional transition probabilities from one state to another, $\mathcal{E}$ are the conditional observation probabilities, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, which maps a state and action to a scalar reward that the agent receives.

Case-based reasoning. Case-based reasoning (CBR) is the process of solving new problems based on the solution of previously seen similar problems. Generally, CBR assumes access to a memory that stores past problems (known as cases) and their solutions. When a new problem is encountered, CBR will *(i)* **retrieve** a similar problem and its solution from memory; *(ii)* **reuse** the solution by mapping it to the current problem; *(iii)* **revise** the solution by testing it and checking whether it is a viable way to address the new problem; and *(iv)* **retain** the solution in memory if the adaptation to the new problem was successful.

4.3 Case-based reasoning in reinforcement learning

This section introduces our framework inspired by CBR for improving generalization in TBGs. Even though we situate our work in TBGs, it serves as a good starting point for applying CBR in more general RL settings. We consider an *on-policy* RL agent

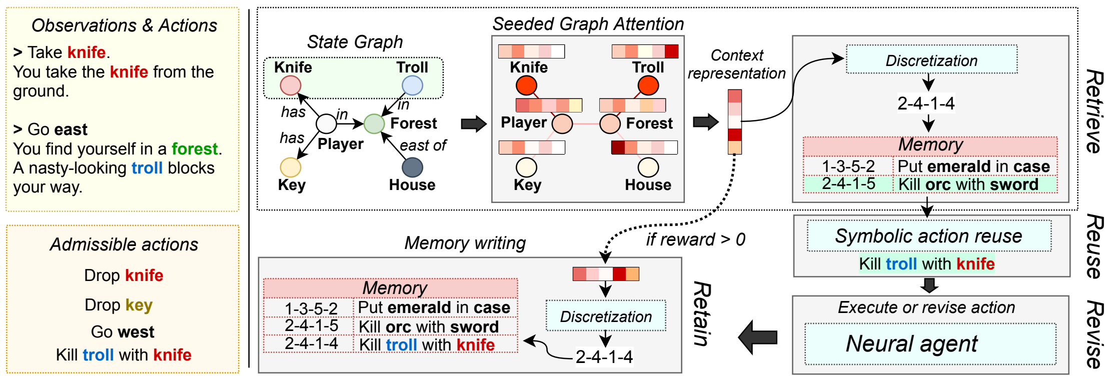


Figure 4.1: Overview of the proposed approach and architecture of the CBR agent. A memory stores actions that have been used successfully in previous interactions. The context of the game is learned from the state knowledge graph using a graph attention mechanism. Actions are retrieved from the memory based on this context representation and mapped to the current state. If no valid action is obtained using CBR, a neural agent explores the most likely action.

that, at any given time step t, has access to a memory $\mathcal{M}_t$, that can be used to retrieve previous experiences. The memory contains key-value pairs, where the keys are a context representation of a game state and values are actions that were taken by the agent w.r.t to this context. As mentioned in Section 4.2, case-based reasoning can be formalized as a four-step process. We describe our proposed methodology for each step below. Algorithm 1 provides a detailed formalization of our approach.

Retrieve. Given the state of the game s_t and the valid actions $\mathcal{A}_t$, we want to retrieve from the memory $\mathcal{M}_t$ previous experiences that might be useful in decision-making at the current state. To this end, for each admissible action $a_t \in \mathcal{A}_t$, we define a context selector $c_t = \textit{context}(s_t, a_t)$. The context selector is an action-specific representation of the state, namely the portion of the state that is relevant to the execution of an action. We will explain later how the context selector is defined in our implementation. For each context c_t, we retrieve from the memory the context-action pair $(c_t^{\mathcal{M}}, a_t^{\mathcal{M}})$, such that $c_t^{\mathcal{M}}$ has maximum similarity with c_t. We denote as $\delta = \textit{sim}(c_t, c_t^{\mathcal{M}}) \in [0, 1]$ the relevance score given to the retrieved action. Only actions $a_t^{\mathcal{M}}$ with a relevance score above a *retriever threshold* τ are retrieved from $\mathcal{M}_t$. We denote as $\mathcal{A}_t^{\mathcal{M}}$ the final set of action-relevance pairs returned by the retriever, as shown in Algorithm 1.

Reuse. The goal of the reuse step is to adapt the actions retrieved from the memory based on the current state. This is accomplished by a *reuse* function, that is applied to each retrieved action to construct a set $\tilde{\mathcal{A}}_t$ of candidate actions that should be applicable to the current state, each paired with a confidence level.

Revise. If any of the action candidates $\tilde{\mathcal{A}}_t$ is a valid action, then the one with the highest relevance δ is executed, otherwise a neural agent π is used to select the best action $a_t^\star$. We denote with $r_t = r(s_t, a_t^\star)$ the obtained reward. Note that π can be an existing agent for TBGs (Keerthiram Murugesan, Atzeni, Kapanipathi, Talamadupula, et al.

Algorithm 1: CBR in Text-based RL

```
• Retrieve
Let $\mathcal{C}_t = \{\textit{context}(s_t, a_t) \mid a_t \in \mathcal{A}_t\}$ be a set of context selectors for state $s_t$ at
  time step $t$
$\mathcal{A}_t^{\mathcal{M}} \leftarrow \emptyset$
for $c_t \in \mathcal{C}_t$ do
    Let $(c_t^{\mathcal{M}}, a_t^{\mathcal{M}}) = \arg\max_{(c_t^{\mathcal{M}}, a_t^{\mathcal{M}}) \in \mathcal{M}_t} \textit{sim}(c_t, c_t^{\mathcal{M}})$
    Let $\delta = \textit{sim}(c_t, c_t^{\mathcal{M}})$
    if $\delta > \tau$ then
        $\mathcal{A}_t^{\mathcal{M}} \leftarrow \mathcal{A}_t^{\mathcal{M}} \cup \{(a_t^{\mathcal{M}}, \delta)\}$
    end
end

• Reuse
Build a set of action candidates:
  $\tilde{\mathcal{A}}_t = \{\textit{reuse}(a_t^{\mathcal{M}}, s_t, \delta) \mid (a_t^{\mathcal{M}}, \delta) \in \mathcal{A}_t^{\mathcal{M}}\}$

• Revise
if $\mathcal{A}_t \cap \tilde{\mathcal{A}}_t \neq \emptyset$ then
    Let $a_t^\star, \delta^\star = \arg\max_{\tilde{a}_t, \delta \in \tilde{\mathcal{A}}_t} \delta$
else
    $a_t^\star = \arg\max_{a_t \in \mathcal{A}_t} \pi(a_t|s_t)$
end
Let $r_t = r(s_t, a_t^\star)$ be the reward obtained at time step $t$ by executing action $a_t^\star$

• Retain
Let $c_t^\star = \textit{context}(s_t, a_t^\star)$ be the context of action $a_t^\star$
$\mathcal{T} = \{(c_t^\star, a_t^\star), \ldots, (c_{t-m+1}^\star, a_{t-m+1}^\star)\}$
Let $\tilde{\mathcal{T}}$ be a set of $k$ pairs sampled from $\mathcal{T}$ according to a retain probability
if $r_t > 0$ then
    $\mathcal{M}_{t+1} \leftarrow \mathcal{M}_t \cup \tilde{\mathcal{T}}$
end
```

2021; Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021; Ammanabrolu and M. J. Hausknecht 2020).

Retain. Finally, the retain step stores successful experiences as new cases in the memory, so that they can be retrieved in the future. In principle, this can be accomplished by storing actions for which the agent obtained positive rewards. However, we found that storing previous actions as well can result in improved performance. Whenever $r_t > 0$, a *retain* function is used to compute the probability of each previous context-action pair to be stored in the memory and a set of k past pairs are sampled and stored accordingly. In our experiments, the *retain* function selects the k most recent actions, but other implementations are possible, as discussed in Section 4.9.

4.4 A CBR policy agent to generalize in text-based games

Designing an agent that can act efficiently in TBGs using the described approach poses several challenges. Above all, efficient memory use is crucial to making the approach practical and scalable. Since the context selectors are used as keys for accessing values in the memory, their representation needs to be such that contexts where similar actions were taken receive similar representations. At the same time, as the state space is exponential, context representations need to be focused only on relevant portions of the state and they need to be compressed and compact.

4.4.1 Representing the context through seeded graph attention

State space as a knowledge graph. Following previous works (Ammanabrolu and Riedl 2019; Ammanabrolu and M. J. Hausknecht 2020; Keerthiram Murugesan, Atzeni, Kapanipathi, Talamadupula, et al. 2021), we represent the state of the game as a dynamic knowledge graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{R}_t, \mathcal{E}_t)$, where a node $v \in \mathcal{V}_t$ represents an entity in the game, $r \in \mathcal{R}_t$ is a relation type, and an edge $v \xrightarrow{r} v' \in \mathcal{E}_t$ represents a relation of type $r \in \mathcal{R}_t$ between entities $v, v' \in \mathcal{V}_t$. In TBGs, the space of valid actions $\mathcal{A}_t$ can be modeled as a template-based action space, where actions a_t are instances of a finite set of templates with a given set of entities, denoted as $\mathcal{V}_{a_t} \subseteq \mathcal{V}_t$. As an example, the action "*kill orc with sword*" can be seen as an instance of the template "*kill* v_1 *with* v_2", where v_1 and v_2 are "*orc*" and "*sword*" respectively.

Seeded graph attention. The state graph $\mathcal{G}_t$ and the entities $\mathcal{V}_{a_t}$ are provided as input to the agent for each action $a_t \in \mathcal{A}_t$, in order to build an action-specific contextualized representation of the state. A pre-trained BERT model (Devlin et al. 2019) is used to get a representation $\mathbf{h}_v^{(0)} \in \mathbb{R}^d$ for each node $v \in \mathcal{V}_t$. Inspired by H. Sun, Dhingra, et al. (2018), we propose a seeded graph attention mechanism (GAT), so that the propagation of messages is weighted more for nodes close to the entities $\mathcal{V}_{a_t}$. Let $\alpha_{vu}^{(l)}$ denote the attention coefficients given by a graph attention network (Velickovic et al. 2018) at layer l for nodes $v, u \in \mathcal{V}_t$. Then, for each node $v \in \mathcal{V}_t$, we introduce a coefficient $\beta_v^{(l)}$ that scales with the amount of messages received by node v at layer l:

$$\beta_v^{(1)} = \begin{cases} \frac{1}{|\mathcal{V}_{a_t}|} & \text{if } v \in \mathcal{V}_{a_t} \\ 0 & \text{otherwise} \end{cases}, \qquad \beta_v^{(l+1)} = (1-\lambda)\beta_v^{(l)} + \lambda \sum_{u \in \mathcal{N}_v} \alpha_{vu}^{(l)} \beta_u^{(l)},$$

where $\mathcal{N}_v$ denotes the neighbors of v, considering the graph as undirected. Note that, at layer $l = 1$, only the nodes in $\mathcal{V}_{a_t}$ receive messages, whereas for increasing values of l, $\beta_v^{(l)}$ will be non-zero for their $(l-1)$-hop neighbors as well. The representation of each $v \in \mathcal{V}_t$ is then updated as:

$$\mathbf{h}_v^{(l)} = \mathit{FFN}^{(l)} \left(\mathbf{h}_v^{(l-1)} + \beta_v^{(l)} \sum_{u \in \mathcal{N}_v} \alpha_{vu}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)} \right),$$

where $\mathit{FFN}^{(l+1)}$ is a 2-layer feed-forward network with ReLU non-linearity and $\mathbf{W}^{(l)} \in \mathbb{R}^{d \times d}$ are learnable parameters. Finally, we compute a continuous contextualized representation $\mathbf{c}_{a_t}$ of the state by summing the linear projections of the hidden representations of each $v \in \mathcal{V}_{a_t}$ and passing the result through a feed-forward network.

4.4.2 Memory access through context quantization

Given a continuous representation $\mathbf{c}_{a_t}$ of the context, we need an efficient way to access the memory $\mathcal{M}_t$ to retrieve or store actions based on such a context selector. Storing and retrieving based on the continuous representation $\mathbf{c}_{a_t}$ would be impractical for scalability reasons. Additionally, since the parameters of the agent change over the training time, the same context would result in several duplicated entries in the memory even with a pre-trained agent over different episodes.

Discretization of the context. To address these problems, we propose to use vector quantization (Ballard 2000) before reading or writing to memory. Following previous work (T. Chen et al. 2018; Sachan 2020), we learn a discretization function $\phi : \mathbb{R}^d \to \mathbb{Z}_K^D$, that maps the continuous representation $\mathbf{c}_{a_t}$ into a K-way D-dimensional code $c_t \in \mathbb{Z}_K^D$, with $|\mathbb{Z}_K| = K$ (we refer to c_t as a KD code). With reference to Section 4.3, then we will use $c_t = \mathit{context}(s_t, a_t) = \phi(\mathbf{c}_{a_t})$ as the context selector used to access the memory $\mathcal{M}_t$. In order to implement the discretization function, we define a set of K key vectors $\mathbf{k}_i \in \mathbb{R}^d, i = 1, \ldots, K$, and we divide each vector in D partitions $\mathbf{k}_i^j \in \mathbb{R}^{d/D}, j = 1, \ldots, D$. Similarly, we divide $\mathbf{c}_{a_t}$ in D partitions $\mathbf{c}_{a_t}^j \in \mathbb{R}^{d/D}, j = 1, \ldots, D$. Then, we compute the j-th code z^j of c_t by nearest neighbor search, as $z^j = \arg\min_i \|\mathbf{c}_{a_t}^j - \mathbf{k}_i^j\|_2^2$. We use the straight-through estimator (Bengio, Léonard, et al. 2013) to address the non differentialbility of the *argmin* operator.

Memory access. The KD codes introduced above are used to provide a memory-efficient representation of the keys in the memory. Then, given the KD code representing the current context selector c_t, we query the memory by computing a similarity measure $\mathit{sim}(c_t, c_t^{\mathcal{M}})$ between c_t and each $c_t^{\mathcal{M}}$ in $\mathcal{M}_t$. The similarity function is defined as the fraction of codes shared by c_t and $c_t^{\mathcal{M}}$. The context-action pair with the highest similarity is returned as a result of the memory access, together with a relevance score δ representing the value of the similarity measure.

4.4.3 Symbolic action reuse and revise policy

We use a simple purely symbolic *reuse* function to adapt the actions retrieved from the memory to the current state. Let c_t be the context selector computed based on state s_t and the entities $\mathcal{V}_{a_t}$, as explained in Sections 4.4.1 and 4.4.2. Denote with $(c_t^{\mathcal{M}}, a_t^{\mathcal{M}})$ the context-action pair retrieved from $\mathcal{M}_t$ with confidence δ. Then, the reuse function *reuse*$(a_t^{\mathcal{M}}, s_t, \delta)$ constructs the action candidate $\tilde{a}_t$ as the action with the same template as $a_t^{\mathcal{M}}$ applied to the entities $\mathcal{V}_{a_t}$. If the reuse step cannot generate a valid action, we revert to the neural policy agent π that outputs a probability distribution over the current admissible actions $\mathcal{A}_t$.

4.5 Training

In Section 4.3, we have introduced an on-policy RL agent that relies on case-based reasoning to act in the world efficiently. This agent can be trained in principle using any online RL method. This section discusses the training strategies and learning objectives used in our implementation.

Objective. Two main portions of the model need to be trained: *(a)* the *retriever*, namely the neural network that computes the context representation and accesses the memory through its discretization, and *(b)* the main neural agent π which is used in the revise step. Note that π can be any on-policy agent designed for text-based games (Ammanabrolu and M. J. Hausknecht 2020; Keerthiram Murugesan, Atzeni, Kapanipathi, Talamadupula, et al. 2021; Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021). All agents π used in our experiments are trained with an Advantage Actor-Critic (A2C) method. For optimizing the parameters of π, we use the same learning objectives defined by Adolphs et al. (2019), as explained below. Whenever the executed action $a_t^\star$ is not chosen by the model π but it comes from the symbolic reuse step, then we optimize instead an additional objective for the retriever, namely the following contrastive loss (Hadsell et al. 2006):

$$\mathcal{L}_r^{(t)} = \frac{1}{2}(1 - y_t)(1 - \mathit{sim}(c_t, c_t^{\mathcal{M}}))^2 + \frac{1}{2} y_t \max\{0, \mu - 1 + \mathit{sim}(c_t, c_t^{\mathcal{M}})\}^2,$$

where c_t denotes the context selector of the action executed at time step t, $c_t^{\mathcal{M}}$ is the corresponding key entry retrieved from $\mathcal{M}_t$, μ is the margin parameter of the contrastive loss, and $y_t = 1$ if $r_t > 0$, $y_t = 0$ otherwise. This objective encourages the retriever to produce similar representations for two contexts where reusing an action yielded a positive reward.

Pretraining. To make learning more stable and allow the agent to act more efficiently, we found it beneficial to pretrain the retriever. This minimizes large shifts in the context representations over the training time. We run a baseline agent (Ammanabrolu and

M. J. Hausknecht 2020) to collect instances of the state graph and actions that yielded positive rewards. Then we train the retriever to encode with similar representations the contexts for which similar actions (i.e., actions with the same template) were used. This is achieved using the same contrastive loss defined above.

Training details. All agents used in our experiments are trained with an Advantage Actor Critic (A2C) method according to the general scheme defined below. We use the n-step temporal difference method (Sutton et al. 2018) to compute the return $R(s_t, a_t)$ of a single time step t in a session of length T as:

$$R(s_t, a_t) = \gamma^{T-t} V(s_T) + \sum_{i=0}^{T-t} \gamma^i r(s_{t+i}, a_{t+i}),$$

where $V(s_T)$ denotes the value of s_T computed by the critic network, γ is the discount factor, and r is the reward function. We then compute the advantage $A(s_t, a_t) = R(s_t, a_t) - V(s_t)$. The final objective term consists of four separate objectives. First, we have $\mathcal{L}_\pi^{(t)}$ that denotes the objective of the policy, which tries to maximize the the advantage A:

$$\mathcal{L}_\pi^{(t)} = -A(s_t, a_t^\star) \log \pi(a_t^\star | s_t).$$

We then add the objective of the critic as:

$$\mathcal{L}_v^{(t)} = \frac{1}{2} \left(R(s_t, a_t^\star) - V(s_t) \right)^2 .$$

$\mathcal{L}_v^{(t)}$ encourages the value of the critic V to better estimate the reward R by reducing the mean squared error between them. To prevent the policy from assigning a large weight on a single action, we perform entropy regularization by introducing an additional term:

$$\mathcal{L}_e^{(t)} = \eta \cdot \sum_{a_t \in \mathcal{A}_t} \pi(a_t | s_t) \cdot \log \pi(a_t | s_t).$$

The η parameter helps balance the exploration-exploitation trade-off for the policy. The sum of the above objectives defines the loss when the neural agent π is used to select the action $a_t^\star$. In case actions are reused from the memory, then we use the contrastive loss as discussed in Section 4.5, namely we compute the loss as:

$$\mathcal{L}_r^{(t)} = \frac{1}{2}(1 - y_t)(1 - \mathit{sim}(c_t, c_t^{\mathcal{M}}))^2 + \frac{1}{2} y_t \max\{0, \mu - 1 + \mathit{sim}(c_t, c_t^{\mathcal{M}})\}^2.$$

4.6 Main experimental results

This section provides a detailed evaluation of our approach. We assess quantitatively the performance of CBR combined with existing RL approaches and we demonstrate its capability to improve sample efficiency and generalize out of the training distribution.

	Easy		Medium		Hard	
	#Steps	**Norm. Score**	**#Steps**	**Norm. Score**	**#Steps**	**Norm. Score**
Text	23.83 ± 2.16	0.88 ± 0.04	44.08 ± 0.93	0.60 ± 0.02	49.84 ± 0.38	0.30 ± 0.02
TPC	20.59 ± 5.01	0.89 ± 0.06	42.61 ± 0.65	0.62 ± 0.03	48.45 ± 1.13	0.32 ± 0.04
KG-A2C	22.10 ± 2.91	0.86 ± 0.06	41.61 ± 0.37	0.62 ± 0.03	48.00 ± 0.61	0.32 ± 0.00
BiKE	18.27 ± 1.13	0.94 ± 0.02	39.34 ± 0.72	0.64 ± 0.02	47.19 ± 0.64	0.34 ± 0.02
CBR-only	22.13 ± 1.98	0.80 ± 0.05	43.76 ± 1.23	0.62 ± 0.03	48.12 ± 1.30	0.33 ± 0.06
Text + CBR	17.53 ± 3.36	0.93 ± 0.04	39.10 ± 1.77	0.66 ± 0.04	47.11 ± 1.21	0.34 ± 0.02
TPC + CBR	16.81 ± 3.12	0.94 ± 0.03	37.05 ± 1.61	0.67 ± 0.03	47.25 ± 1.56	0.37 ± 0.03
KG-A2C + CBR	15.91 ± 2.52	0.95 ± 0.03	36.13 ± 1.65	0.66 ± 0.05	46.11 ± 1.13	0.40 ± 0.04
BiKE + CBR	15.72 ± 1.15	0.95 ± 0.04	35.24 ± 1.22	0.67 ± 0.03	45.21 ± 0.87	0.42 ± 0.04

Table 4.1: Test-set performance for *TWC in-distribution* games

Next, we provide qualitative insights and examples of the behavior of the model and we perform an ablation study to understand the role played by the different components of the architecture.

4.6.1 Experimental setup

Agents. We consider several agents obtained by plugging existing RL methods in the revise step. We first define two simple approaches: **CBR-only**, where we augment a random policy with the CBR approach, and **Text + CBR**, which relies on the CBR method combined with a simple GRU-based policy network that consumes as input the textual observation from the game. Next, we select three recently proposed TBG approaches: **Text+Commonsense** (**TPC**, the agent of Chapter 2) (Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021), **KG-A2C** (Ammanabrolu and M. J. Hausknecht 2020), and **BiKE** (Keerthiram Murugesan, Atzeni, Kapanipathi, Talamadupula, et al. 2021) (the agent of Chapter 3, to create the **TPC + CBR**, **KG-A2C + CBR** and **BiKE + CBR** agents. We consider the original agents that are not trained with CBR as baselines.

Datasets. We empirically verify the efficacy of our approach on **TextWorld Commonsense** (*TWC*) (Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021) and **Jericho** (M. Hausknecht et al. 2020). *Jericho* is a well-known and challenging learning environment including 33 interactive fiction games. *TWC* is an environment which builds on *TextWorld* (Côté et al. 2018) and provides a suite of games requiring commonsense knowledge. *TWC* allows agents to be tested on two settings: the *in-distribution games*, where the objects that the agent encounters in the test set are the same as the objects in the training set, and the *out-of-distribution games* which have no entity in common with the training set. For each of these settings, *TWC* provides three difficulty levels: *easy*, *medium*, and *hard*.

Evaluation metrics. Following the same experimental setup of Chapter 2 (Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021), we evaluate the agents on *TWC*

	Easy		Medium		Hard	
	#Steps	**Norm. Score**	**#Steps**	**Norm. Score**	**#Steps**	**Norm. Score**
Text	29.90 ± 2.92	0.78 ± 0.02	45.90 ± 0.22	0.55 ± 0.01	50.00 ± 0.00	0.20 ± 0.02
TPC	27.74 ± 4.46	0.78 ± 0.07	44.89 ± 1.52	0.58 ± 0.01	50.00 ± 0.00	0.19 ± 0.03
KG-A2C	28.34 ± 3.63	0.80 ± 0.07	43.05 ± 2.52	0.59 ± 0.01	50.00 ± 0.00	0.21 ± 0.00
BiKE	25.59 ± 1.92	0.83 ± 0.01	41.01 ± 1.61	0.61 ± 0.01	50.00 ± 0.00	0.23 ± 0.02
CBR-only	23.43 ± 2.09	0.80 ± 0.04	44.03 ± 1.75	0.63 ± 0.04	48.71 ± 1.15	0.31 ± 0.03
Text + CBR	20.91 ± 1.72	0.89 ± 0.02	40.32 ± 1.27	0.66 ± 0.04	47.89 ± 0.87	0.32 ± 0.06
TPC + CBR	18.90 ± 1.91	0.92 ± 0.01	37.30 ± 1.00	0.66 ± 0.02	47.54 ± 1.67	0.34 ± 0.03
KG-A2C + CBR	18.21 ± 1.32	0.90 ± 0.02	37.02 ± 1.22	0.68 ± 0.03	47.10 ± 1.12	0.38 ± 0.02
BiKE + CBR	17.15 ± 1.45	0.93 ± 0.03	35.45 ± 1.40	0.67 ± 0.03	45.91 ± 1.32	0.40 ± 0.03

Table 4.2: Test-set performance for *TWC out-of-distribution* games

based on the number of steps (**#Steps**) required to achieve the goal (lower is better) and the normalized cumulative reward (**Norm. Score**) obtained by the agent (larger is better). On *Jericho*, we follow previous work (M. Hausknecht et al. 2020; X. Guo et al. 2020; Ammanabrolu and M. J. Hausknecht 2020) and we report the average score achieved over the last 100 training episodes.

4.6.2 Results on TextWorld Commonsense

Table 4.1 reports the results on *TWC* for the *in-distribution* set of games. Overall, we observe that CBR consistently improves the performance of all the baselines. The performance boost is large enough that even a simple method as **Text + CBR** outperforms all considered baselines except **BiKE**.

Out-of-distribution generalization. CBR's ability to retrieve similar cases should allow our method to better generalize to new and unseen problems. We test this hypothesis on the *out-of-distribution* games in *TWC*. The results of this experiment are reported in Table 4.2. We notice that all existing approaches fail to generalize out of the training distribution and suffer a substantial drop in performance in this setting. However, when coupled with CBR, the drop is minor (on average **6%** with CBR *vs* **35%** without on the hard level). Interestingly, even the **CBR-only** agent achieves competitive results compared to the top-performing baselines.

Sample efficiency. Another key benefit of our approach comes as better sample efficiency. With its ability to explicitly store prior solutions effectively, CBR allows existing algorithms to learn faster. Figure 4.2 shows the learning curves for our best agents and the corresponding baselines. The plots report the performance of the agent over the training episodes, both in terms of the number of steps and the normalized score. Overall, we observe that the CBR agents obtain faster convergence to their counterparts on all difficulty levels.

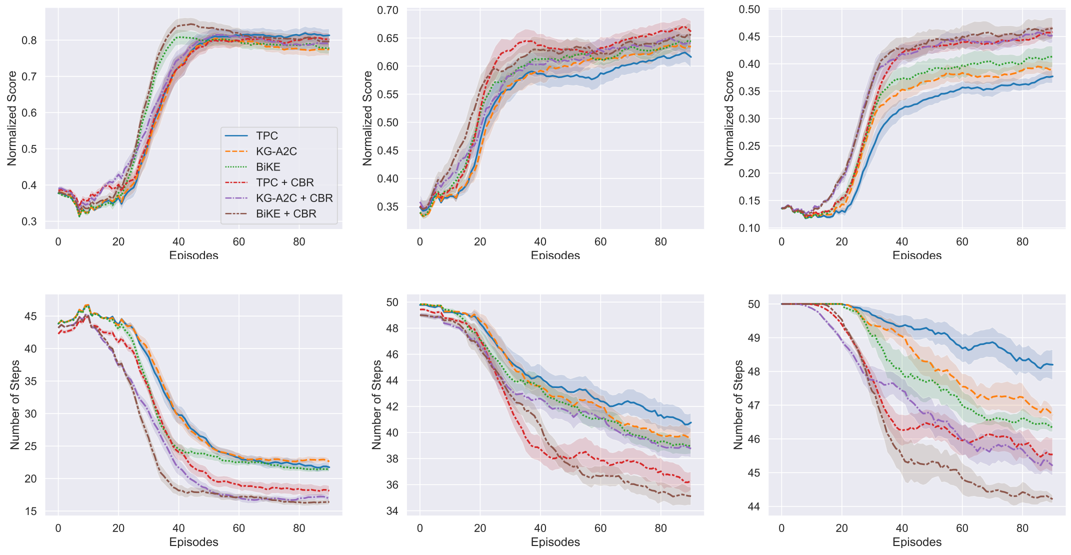


Figure 4.2: Performance on *TWC* (showing mean and standard deviation averaged over 5 runs) for the three difficulty levels: *easy* (left), *medium* (middle), *Hard* (right) using normalized score and number of steps.

4.6.3 Performance on the Jericho games

We evaluate our best performing variant from the experiments on *TWC* (**BiKE + CBR**) against existing approaches on the 33 games in the *Jericho* environment. We compare our approach against strong baselines, including **TDQN** (M. Hausknecht et al. 2020), **DRRN** (J. He et al. 2016), **KG-A2C** (Ammanabrolu and M. J. Hausknecht 2020), **MPRC-DQN** (X. Guo et al. 2020), and **RC-DQN** (X. Guo et al. 2020). The same experimental setting and handicaps as the baselines are used, as we train for $100\,000$ steps and we assume access to valid actions. Table 4.3 summarizes the results of the *Jericho* games. We observe that our CBR agent achieves comparable or better performance than any baseline on 24 (73%) of the games, strictly outperforming all the other agents in 18 games, thereby setting new state-of-the-art results on *Jericho*.

4.6.4 Qualitative analysis and ablation studies

Insights on the model. Figure 4.3 provides two examples showing the **BiKE + CBR** agent interacting with the **zork1** game. In the example on top, the agent retrieves an experience that can be successfully reused and turned into a valid action at the current time step. The heat maps visualize the value of the context similarity function defined in Section 4.4 for the top entries in the memory. In the negative example at the bottom instead, the agent retrieves an action that is not useful and needs to fall back to the neural policy π. Figure 4.4 (top) shows the fraction of times that actions retrieved from the memory are reused successfully in the *TWC* games. We observe that, both for *in-distribution* and *out-of-distribution* games, the trained agent relies on

Game	Human (max)	Human (Walkthrough-100)	TDQN	DRRN	KG-A2C	MPRC-DQN	RC-DQN	BiKE + CBR
905	1	1	**0**	**0**	**0**	**0**	**0**	**0**
acorncourt	30	30	1.6	10	0.3	10	10	**12.2**
adventureland	100	42	0	20.6	0	24.2	21.7	**27.3**
afflicted	75	75	1.3	2.6	–	**8**	**8**	3.2
awaken	50	50	**0**	**0**	**0**	**0**	**0**	**0**
detective	360	350	169	197.8	207.9	317.7	291.3	**326.1**
dragon	25	25	-5.3	-3.5	0	0.04	4.84	**8.3**
inhumane	90	70	0.7	0	3	0	0	**24.2**
library	30	30	6.3	17	14.3	17.7	18.1	**22.3**
moonlit	1	1	**0**	**0**	**0**	**0**	**0**	**0**
omniquest	50	50	16.8	10	3	10	10	**17.2**
pentari	70	60	17.4	27.2	50.7	44.4	43.8	**52.1**
reverb	50	50	0.3	**8.2**	–	2	2	6.5
snacktime	50	50	9.7	0	0	0	0	**22.1**
temple	35	20	7.9	7.4	7.6	**8**	**8**	7.8
ztuu	100	100	4.9	21.6	9.2	85.4	79.1	**87.2**
advent	350	113	36	36	36	**63.9**	36	62.1
balances	51	30	4.8	10	10	10	10	**11.9**
deephome	300	83	**1**	**1**	**1**	**1**	**1**	**1**
gold	100	30	**4.1**	0	–	0	0	2.1
jewel	90	24	0	1.6	1.8	4.46	2	**6.4**
karn	170	40	0.7	2.1	0	**10**	**10**	0
ludicorp	150	37	6	13.8	17.8	19.7	17	**23.8**
yomomma	35	34	0	0.4	–	**1**	**1**	**1**
zenon	20	20	0	0	3.9	0	0	**4.1**
zork1	350	102	9.9	32.6	34	38.3	38.8	**44.3**
zork3	7	3	0	0.5	0.1	**3.63**	2.83	3.2
anchor	100	11	**0**	**0**	**0**	**0**	**0**	**0**
enchanter	400	125	8.6	20	12.1	20	20	**36.3**
sorcerer	400	150	5	20.8	5.8	**38.6**	38.3	24.5
spellbrkr	600	160	18.7	37.8	21.3	25	25	**41.2**
spirit	250	8	0.6	0.8	1.3	3.8	**5.2**	4.2
tryst205	350	50	0	9.6	–	10	10	**13.4**
Best agent			6 (18%)	6 (18%)	5 (15%)	12 (36%)	10 (30%)	24 (73%)

Table 4.3: Average raw score on the *Jericho* games. We denote with colors the difficulty of the games (green for *possible* games, yellow for difficult games and red for extreme games). The last row reports the fraction and the absolute number of games where an agent achieves the best score. We additionally report human performance (**Human – max**) and the 100-step results from a human-written walkthrough (**Human – Walkthrough 100**). Results are taken from the original papers or "–" is used if a result was not reported.

CBR from 60% to approximately 70% of the times. Figure 4.4 (bottom) further shows the fraction of times that the neural agent would have been able to select a rewarded action as well, when the CBR reuses a successful action. The plot shows that, for the *out-of-distribution* games, the neural agent would struggle to select good actions when the CBR is used.

Main ablation studies. In order to understand the role of the main modules of our CBR agent, we designed some ablation studies. First, instead of using the seeded GAT, we define the context of a state-action pair *context*(s_t, a_t) as just one of the entities that a_t is applied to. This definition suits well the *TWC* games because rewarded actions are always applied to one target object and a location for that object (see Appendix B.1 for details). Note that, since the set of entities is discrete, no context quantization is

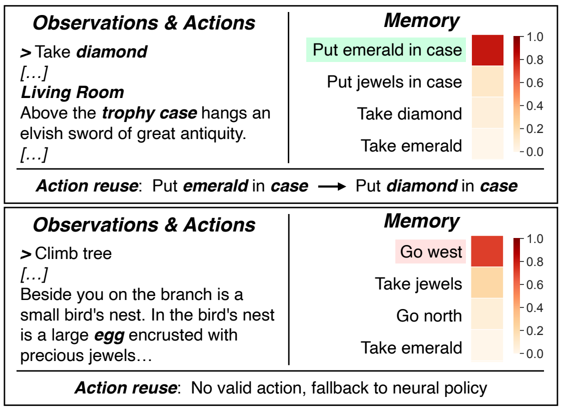

Observations & Actions
> Take diamond
[...]
Living Room
Above the trophy case hangs an elvish sword of great antiquity.
[...]
Memory
Put emerald in case
Put jewels in case
Take diamond
Take emerald
1.0
0.8
0.6
0.4
0.2
0.0
Action reuse: Put emerald in case → Put diamond in case
Observations & Actions
> Climb tree
[...]
Beside you on the branch is a small bird's nest. In the bird's nest is a large egg encrusted with precious jewels...
Memory
Go west
Take jewels
Go north
Take emerald
Action reuse: No valid action, fallback to neural policy

Figure 4.3: Examples from the **zork1** game, showing the content of the memory and the context similarities, in a situation where the agent is able to reuse a previous experience and in a case where the revise step is needed.

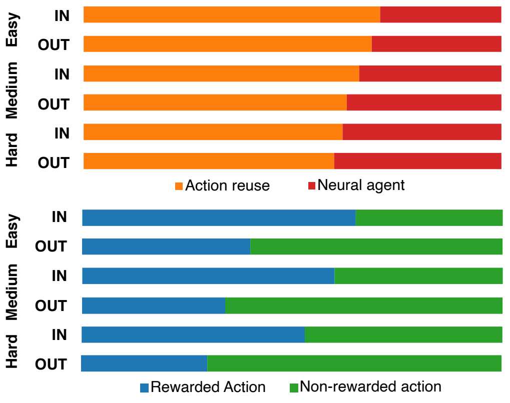


Figure 4.4: Fraction of times that a retrieved action is reused successfully on *TWC* (top). Fraction of times that the neural agent would have picked a rewarded action when CBR is used successfully (bottom).

		Easy	Medium	Hard
IN	**BiKE + CBR (w/o GAT)**	16.32 ± 1.10	36.13 ± 1.40	45.72 ± 0.63
	BiKE + CBR (w/o VQ)	22.67 ± 1.23	43.18 ± 2.10	49.21 ± 0.55
OUT	**BiKE + CBR (w/o GAT)**	18.15 ± 1.51	37.10 ± 1.41	46.70 ± 0.71
	BiKE + CBR (w/o VQ)	27.75 ± 2.11	44.55 ± 1.67	50.00 ± 0.00

Table 4.4: Results of the ablation study on *TWC*, evaluated based on the number of steps (**#Steps**) to solve the games.

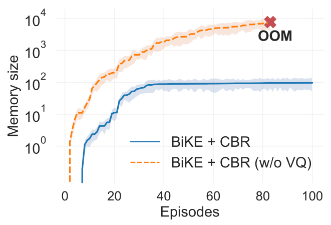


Figure 4.5: Number of entries in the memory over training.

needed. We report the performance of the resulting **BiKE + CBR (w/o GAT)** agent in Table 4.4. The results show that CBR on *TWC* is effective even with this simple context definition, but the lower performance of the agent demonstrates the advantage of incorporating additional context information. Finally, we investigate the role played by vector quantization, by experimenting with an agent (**BiKE + CBR w/o VQ**) that stores the continuous context representations. In general, this poses scalability challenges, but since *TWC* has only 5 games per difficulty level, each with a small number of objects, we were able to evaluate the performance of this agent on the three levels separately. The results, reported in Table 4.4, show that this agent performs much worse than the other CBR implementations. This happens because storing continuous representations over the training results in duplicated entries in the memory and makes it harder to retrieve meaningful experiences. Figure 4.5 demonstrates how the size (number of entries) in the memory grows over the training time. In this experiment, we trained the agent on all difficulty levels at the same time, resulting in the implementation running out of memory (OOM) on the GPU.

4.7 Enhancing baseline agents with CBR on Jericho

In this section, we report additional experimental results on a subset of the *Jericho* games, in order to show the performance improvement obtained by different baseline agents when enhanced with case-based reasoning. Table 4.5 shows the results obtained when coupling the agents described in Section 4.6.1 with CBR. Similarly to what we discussed for *TWC* in Section 4.6.2, we observe that CBR consistently improves the performance of all the agents. The best performing agent is **BiKE + CBR**, which is the agent that we evaluated on the complete set of games in Section 4.6.3.

Game	KG-A2C	KG-A2C + CBR	Text	Text + CBR	TPC	TPC + CBR	BiKE	BiKE + CBR
detective	207.9	**255.6**	205.8	**242.3**	245.6	**315.1**	278.2	**326.1**
inhumane	3	**15.6**	1.1	**14.5**	4.5	**18.3**	9.2	**24.2**
snacktime	0	**15.5**	8.1	**9.8**	15.7	**19.3**	18.8	**22.1**
karn	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
zork1	34	**34.2**	31.5	**38.2**	36	**39.2**	39.5	**44.3**
zork3	0.1	**1.7**	0	**1.6**	1.7	**1.9**	2.5	**3.2**
enchanter	12.1	**26.2**	10.1	**26.4**	13.2	**24.5**	19.7	**36.3**
spellbrkr	21.3	**36.1**	20.4	**33.2**	38.1	**40.2**	38.8	**41.2**

Table 4.5: Additional results on *Jericho* showing the performance improvement obtained by enhancing several baseline agents with CBR.

4.8 Ablation study on memory access

In this section, we investigate the effectiveness of our approach based on vector quantization for efficient memory access. We consider several variants, where VQ is either dropped completely or replaced with other techniques.

4.8.1 Alternatives to vector quantization

All the agents we consider are variants of the best-performing agent on *Jericho* and *TWC*, namely the **BiKE + CBR** agent. We experiment with the following techniques.

- **BiKE + CBR (w/o VQ)** completely removes the vector quantization and stores the continuous context representations as keys in the case memory. In order to make this feasible, for the experiments on *Jericho*, we limit the size of the memory to the 5000 most recent entries.
- **BiKE + CBR** (**RP**) relies on random projection (RP) in order to reduce the dimensionality of the context representations stored by the CBR approach. In this case, each context representation is projected into a p-dimensional space using a random matrix $R \in \mathbb{R}^{p \times d}$, with components drawn from a normal distribution $N(0, \frac{1}{p})$ with mean 0 and standard deviation $\frac{1}{p}$.

- **BiKE + CBR (SRP)** employs sign random projection (SRP): context representations are projected to a p-dimensional space and then discretized by applying an element-wise sign function.

- **BiKE + CBR (LSH)** replaces vector quantization with locality sensitive hashing (LSH). In this case, context representations are converted into h-bit hash codes for l different hash tables. The retrieve step selects the representation with the highest cosine similarity to the query context encoding, among the vectors falling in the same bucket.

4.8.2 Results and discussion

Table 4.6 shows the scores achieved by each variant of the **BiKE + CBR** method on a subset of the *Jericho* games. We notice that the approaches relying on continuous context representations (**w/o VQ** and **RP**) perform poorly compared to the others and to the plain **BiKE** agent. This confirms our hypothesis that CBR needs discrete context representations to make the retrieve step more stable.

The **BiKE + CBR (LSH)** agent, which relies on locality sensitive hashing, achieves competitive results and consistently outperforms the **BiKE** agent. However, we observe that our **BiKE + CBR** agent, based on vector quantization, achieves better results on almost all the games, confirming the benefit of learning the discrete representation as well. Note also that LSH requires storing the complete continuous representations to be able to detect false positives, namely contexts with the same hash code as the query vector, but low cosine similarity. This makes this alternative less memory efficient compared to our implementation based on VQ.

Game	BiKE + CBR (w/o VQ)	BiKE + CBR (RP)	BiKE + CBR (SRP)	BiKE + CBR (LSH)	BiKE + CBR
detective	205.2	203.1	223.2	319.1	**326.1**
inhumane	1.5	3.2	1.1	20.1	**24.2**
snacktime	9.1	14.3	13.2	20.3	**22.1**
karn	**0**	**0**	**0**	**0**	**0**
zork1	31.5	36.3	40.5	41.2	**44.3**
zork3	0	2.7	2.7	**3.6**	3.2
enchanter	10.2	9.2	10.6	35.3	**36.3**
spellbrkr	21.3	23.8	30.8	39.3	**41.2**

Table 4.6: Ablation study showing the results obtained on a subset of the *Jericho* games when the vector quantization is removed (**w/o VQ**) or replaced with random projection (**RP**), sign random projection (**SRP**) or locality sensitive hashing (**LSH**).

4.9 Ablation study on the retain module

In this section, we evaluate different alternatives to select which actions should be retained in the memory of the agent.

4.9.1 Alternatives for the retain module

In our main experiments described in Section 4.6, the retain module has been implemented to store the last k context-action pairs in the memory, whenever the reward obtained by the agent is positive. For *Jericho*, we set $k = 3$, as specified in Appendix B.3. However, other design choices are possible. We consider the following variants of the **BiKE + CBR** agent.

- **BiKE + CBR (rewarded action only)** retains only the rewarded context-action pair, without sampling any of the previous actions. This variant is a simple implementation that works well in practice, but may fail to identify useful actions that were not rewarded.

- **BiKE + CBR (TD error)** samples previous context-action pairs based on the temporal difference (TD) error. In details, the agent still retains k context-action pairs: whenever the reward r_t at time step t is positive, the rewarded context-action pair $(c_t^\star, a_t^\star)$ is retained, together with $k-1$ additional pairs sampled from the current trajectory, with a probability proportional to the TD error.

4.9.2 Results and discussion

Table 4.7 shows the scores obtained by the agents on a subset of the *Jericho* games. Our main implementation storing the last $k = 3$ actions achieves the best results, whereas the **BiKE + CBR (rewarded action only)** agent performs slightly worse than the other two implementations. The agent based on the TD error achieves competitive results and a state-of-the-art score on the **snacktime** game. We observe that all variants perform well in practice and achieve overall better results than the baselines reported in Table 4.3.

Game	BiKE + CBR (rewarded action only)	BiKE + CBR (TD error)	BiKE + CBR
detective	324.1	324.8	**326.1**
inhumane	20.1	23.5	**24.2**
snacktime	20.3	**23.4**	22.1
karn	**0**	**0**	**0**
zork1	40.2	42.4	**44.3**
zork3	**3.2**	**3.2**	**3.2**
enchanter	35.3	34.1	**36.3**
spellbrkr	40.3	40.8	**41.2**

Table 4.7: Ablation study showing the results obtained on a subset of the *Jericho* games when only the context-action pair achieving positive reward is retained (**rewarded action only**), when context-action pairs are sampled using the TD error (**TD error**), and when the last 3 pairs are retained (**BiKE + CBR**).

4.10 Training the agent and the retriever jointly

In this experiment, we try to understand if training the neural agent and the CBR retriever jointly would improve upon our choice of just training the two networks separately. Our implementation works by training the neural agent π and the CBR retriever separately with different objectives. However, we can make the neural agent aware of the CBR retriever and train the two networks jointly. In this case, we need to modify the architecture of the neural agent π in order to take into account the action candidates $\tilde{\mathcal{A}}_t$ produced by the CBR. In details, we compute an action selector by attention between the valid actions $\mathcal{A}_t$ and the candidates $\tilde{\mathcal{A}}_t$ and we concatenate this action selector to the vector used by the neural agent to score admissible actions. Then, the agent and the retriever can be trained jointly, optimizing an objective given by the sum of all the losses in Section 4.5.

Table 4.8 and 4.9 show the results obtained by the baseline agents when they are trained jointly with the CBR retriever. On *TWC*, we observe that the joint variants of the agents achieve comparable results with their counterparts in Table 4.1 and 4.2. On *Jericho*, the agents trained jointly with the retriever achieve strong results, but they perform slightly worse than our main approach that keeps the retriever separate from the neural agent. This shows that the joint version is slightly harder to train. Also, it brings the disadvantage that the architecture of the neural agent has to be changed to take the CBR into account, whereas our main approach that keeps the retriever separate allows readily plugging any on-policy agent for TBGs.

		Easy		Medium		Hard	
		#Steps	**Norm. Score**	**#Steps**	**Norm. Score**	**#Steps**	**Norm. Score**
IN	**Text + CBR (joint)**	17.91 ± 3.80	0.91 ± 0.04	40.22 ± 1.70	0.65 ± 0.05	47.94 ± 1.10	0.33 ± 0.02
	TPC + CBR (joint)	16.80 ± 1.97	0.94 ± 0.04	36.12 ± 1.32	0.67 ± 0.03	46.10 ± 0.72	0.41 ± 0.03
	KG-A2C + CBR (joint)	16.40 ± 1.89	0.95 ± 0.05	36.50 ± 1.13	0.68 ± 0.03	46.58 ± 0.91	0.40 ± 0.06
	BiKE + CBR (joint)	16.01 ± 1.37	0.94 ± 0.03	35.93 ± 1.11	0.67 ± 0.05	46.11 ± 1.14	0.42 ± 0.04
OUT	**Text + CBR (joint)**	21.47 ± 2.32	0.88 ± 0.07	39.10 ± 1.33	0.67 ± 0.02	48.10 ± 0.92	0.31 ± 0.03
	TPC + CBR (joint)	17.89 ± 1.82	0.93 ± 0.01	38.11 ± 1.33	0.65 ± 0.03	47.92 ± 1.55	0.34 ± 0.03
	KG-A2C + CBR (joint)	18.19 ± 2.12	0.93 ± 0.02	37.72 ± 2.91	0.66 ± 0.03	47.53 ± 1.11	0.40 ± 0.04
	BiKE + CBR (joint)	18.12 ± 1.21	0.94 ± 0.05	35.77 ± 1.05	0.69 ± 0.05	46.16 ± 1.00	0.40 ± 0.04

Table 4.8: Test-set results obtained on *TWC in-distribution* (**IN**) and *out-of-distribution* (**OUT**) games training different neural agents and the CBR retriever jointly.

4.11 Multi-paragraph text-based retriever

This section describes an alternative to our graph-based retriever, which only relies on the textual observations without modeling the state of the game as a graph.

Recent work (X. Guo et al. 2020) has shown that enriching the current observation with relevant observations retrieved from the history of interactions with the environment can achieve competitive results on *Jericho*. Therefore, in order to assess the

Game	**KG-A2C + CBR (joint)**	**Text + CBR (joint)**	**TPC + CBR (joint)**	**BiKE + CBR (joint)**
detective	250.1	241.5	**316.2**	324.2
inhumane	15.1	13.2	16.3	24.1
snacktime	13.2	**11.2**	**20.1**	21.8
karn	0	0	0	0
zork1	**36.4**	36.5	36.3	43.7
zork3	1.5	1.5	1.8	**3.6**
enchanter	25.1	26.1	21.1	35.6
spellbrkr	32.3	**33.3**	39.2	**41.8**

Table 4.9: Results obtained on a subset of the *Jericho* games training different neural agents and the CBR retriever jointly. Bold values indicate when the joint variant achieves better scores than the main counterparts reported in Table 4.5.

effectiveness of our graph-based implementation, we compare to a multi-paragraph text-based retriever (MTPR) inspired by the work of X. Guo et al. (2020). In this case, we do not model the state as a graph and, subsequently, we remove the seeded graph attention mechanism from the retriever. Instead, given the current natural language observation o_t, we compute an action-specific representation following X. Guo et al. (2020), concatenating o_t with the n most recent observations that share objects with it or with the given action. The encoded observation is then discretized using vector quantization as in our main architecture.

We evaluated the text-based retriever on *Jericho*, integrating it in the same baseline agents described in Section 4.6.1. Table 4.10 shows the scores obtained by the agents. We observe that, overall, the graph-based retriever performs better on the vast majority of the games. This result confirms the ability of our approach based on seeded graph attention to extract relevant information from the state of the game, compared to a retriever that only relies on text information.

Game	**KG-A2C + CBR (MPTR)**	**Text + CBR (MPTR)**	**TPC + CBR (MPTR)**	**BiKE + CBR (MPTR)**
detective	245.2	233.7	302.3	321.2
inhumane	13.4	13.2	12.3	20.3
snacktime	12.1	8.2	18.1	19.5
karn	0	0	0	0
zork1	**36.2**	36.2	37.4	42.2
zork3	0.8	1.2	**3.2**	**3.6**
enchanter	19.3	24.3	20.2	32.1
spellbrkr	31.3	30.3	39.3	40.8

Table 4.10: Results obtained on a subset of the *Jericho* games using the multi-paragraph text-based retriever (**MPTR**) instead of the graph-based one. Bold values indicate when the **MPTR** variant achieves better scores than the main counterparts reported in Table 4.5.

4.12 Case-based reasoning and OOD generalization

Our experiments on *TWC* allowed assessing the hypothesis that case-based reasoning can be used to tackle out-of-distribution (OOD) generalization in text-based games. Table 4.11 shows the absolute OOD generalization gap of the different agents evaluated in our experiments. Note that this table does not report any new result, but it simply provides the absolute difference between the values in Table 4.1 and 4.2. We observe that the agents relying on CBR achieve a considerably better generalization performance out of the training distribution, almost comparable to the results obtained on the same distribution as the training data. In some cases, the normalized score achieved by the CBR agents in the out-of-distribution games equals the score obtained in the in-distribution games. This happens because case-based reasoning forces the agent to map contexts including entities that were not seen at training time to the most similar contexts in the CBR memory. CBR allows the agent to solve completely new problems and generalize by effectively retrieving past cases and mapping the retrieved actions from the training distribution to the most similar options in the OOD setting. Note that the only good OOD generalization gap for the agents that are not relying on CBR (the **#Steps** of the **Text** agent on the **Hard** level) is an artifact of the experiment, as all agents were limited to a maximum of 50 steps.

Figure B.1 shows a nice example of the capability of the agent to generalize OOD. In this case, entity embeddings were used as the context representations, and we observe that entities that are not included in the training distribution are correctly mapped to the right cluster. This shows that the CBR approach learns effective and generalizable context representations based on the objective of the games. These representations are then used by the agent to select relevant experiences and map the actions used at training time to the most viable alternative in the OOD test set.

	Easy		**Medium**		**Hard**	
	#Steps	**Norm. Score**	**#Steps**	**Norm. Score**	**#Steps**	**Norm. Score**
Text	6.07	0.10	1.82	0.05	0.16	0.10
TPC	7.15	0.11	2.28	0.04	1.55	0.13
KG-A2C	6.24	0.06	1.44	0.03	2.00	0.11
BiKE	7.32	0.11	1.67	0.03	2.81	0.11
CBR-only	1.30	0.00	0.27	0.01	0.59	0.02
Text + CBR	3.38	0.04	1.22	0.00	0.78	0.02
TPC + CBR	2.09	0.02	0.25	0.01	0.29	0.03
KG-A2C + CBR	2.30	0.05	0.89	0.02	0.99	0.02
BiKE + CBR	1.43	0.02	0.21	0.00	0.70	0.02

Table 4.11: Absolute out-of-distribution generalization gap in *TWC*

4.13 Related work

We have already covered a survey of text-based reinforcement learning in Section 2.5. In the context of RL, CBR has been used to speed up and improve transfer learning in heuristic-based RL. Celiberto Jr et al. (2011) and Bianchi, Santos, et al. (2018) have shown that cases collected from one domain can be used as heuristics to achieve faster convergence when learning an RL algorithm on a different domain. In contrast to these works, we present a scalable way of using CBR alongside deep RL methods in settings with very large state spaces. More recently, CBR has been successfully applied in the field of knowledge-based reasoning. Das, Godbole, et al. (2020) and Das, Zaheer, et al. (2021) show that CBR can effectively learn to generate new logical reasoning chains from prior cases, to answer questions on knowledge graphs.

4.14 Conclusion

In this work, we proposed new agents for TBGs using case-based reasoning. In contrast to expensive deep RL approaches, CBR simply builds a collection of its past experiences and uses the ones relevant to the current situation to decide upon its next action in the game. Our experiments showed that CBR when combined with existing RL agents can make them more efficient and aid generalization in out-of-distribution settings. Even though CBR was quite successful in the TBGs explored in our work, future work is needed to understand the limitations of CBR in such settings.

Factual reasoning in language-understanding tasks

Part II

5 Scaling KBQA by decoupling mulit-hop and logical reasoning

5.1 Introduction

Enhancing machine learning models with the ability to reason over structured data has been a major challenge (Marcus 2020; Lake and Baroni 2018) and has historically required complex systems made of several hand-crafted or learned components (Yao et al. 2014; Ferrucci et al. 2010). Recently, the paradigm has shifted to deep learning approaches (H. Sun, Arnold, et al. 2020; H. Sun, Bedrax-Weiss, et al. 2019), where neural networks are used to reason over structured knowledge or a text corpus. In this work, we assume that the source of knowledge is a structured knowledge graph (KG) and we tackle the problem of *knowledge-based question answering* (KBQA), namely finding answers to natural language queries involving multi-hop and logical reasoning over the KG.

Answering queries over a knowledge graph involves many challenges, among which scalability is a major issue. Real-world KGs often contain millions of nodes and even a 2-hop neighborhood of the entities mentioned in the query may comprise tens of thousands of nodes. Many state-of-the-art approaches (H. Sun, Dhingra, et al. 2018; H. Sun, Bedrax-Weiss, et al. 2019; Saxena et al. 2020) address the challenge of scalability by building small query-dependent subgraphs. To this end, they usually use simple heuristics (H. Sun, Dhingra, et al. 2018) or, in some cases, iterative procedures based on learned classifiers (H. Sun, Bedrax-Weiss, et al. 2019). This preprocessing step is usually needed because each forward pass in end-to-end neural networks for KBQA scales at least linearly with the number of edges in the subgraph. Training neural networks involves repeated evaluation, which renders even a linear complexity impractical for graphs of more than a few tens thousands of nodes.

In order to address this issue, we introduce a novel approach called **SQALER** (Scaling Question Answering by Leveraging Edge Relations). The method first learns a model that generates a set of candidate answers (entities in the KG) by *multi-hop reasoning*:

the candidate solutions are obtained by starting from the set of entities mentioned in the question and seeking those that provide an answer by chained relational following operations. We refer to this module as the *relation-level* model. We show that this multi-hop reasoning step can be done efficiently and provably generates a set of candidates including all the actual answers to the original question. **SQALER** then uses a second-stage *edge-level* model that recovers the real answers by performing logical reasoning on a subgraph in the vicinity of the candidate solutions. A visual summary of our approach is depicted in Figure 5.1.

The main contributions and takeaway messages of this work are the following:

1. KBQA can be addressed by first performing multi-hop reasoning on the KG and then refining the result with more sophisticated logical reasoning without losing expressive power (we will elaborate this claim in more details in Section 5.2.3).
2. Multi-hop reasoning can be accomplished efficiently with a method that scales linearly with the number of relation types in the KG, which are usually significantly fewer than the number of facts or entities.

In the remainder of the chapter, we first provide an extensive overview of our approach and a theoretical analysis of the expressive power and the computational complexity of **SQALER**. Our experimental results show that **SQALER** achieves better reasoning performance than state-of-the-art approaches, generalizes compositionally out of the training distribution, and scales to the size of real-world knowledge graphs with millions of entities.

5.2 Scaling KBQA with relation and edge-level reasoning

This section provides a detailed description of our approach. We start by defining the problem formally and giving an intuitive overview of **SQALER**. Then, we discuss the approach in more details and we analyze both its computational complexity and expressive power.

Problem statement. We denote a knowledge graph as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$, where $v \in \mathcal{V}$ represents an entity or node in $\mathcal{G}$, $r \in \mathcal{R}$ is a relation type, and we write $v \xrightarrow{r} v'$ to denote an edge in $\mathcal{E}$ labeled with relation type $r \in \mathcal{R}$ between two entities $v, v' \in \mathcal{V}$. We extend the same notation to sets of nodes by writing $\mathcal{V}_i \xrightarrow{r} \mathcal{V}_j$ if $\mathcal{V}_j = \{v_j \in \mathcal{V} \mid v_i \xrightarrow{r} v_j, v_i \in \mathcal{V}_i\}$. Given a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ and a natural language question Q, expressed as a sequence of tokens $Q = (q_1, q_2, \ldots, q_{|Q|})$, in *knowledge-based question answering* the objective is to identify a set of nodes $\mathcal{A}_Q \subseteq \mathcal{V}$ representing the correct answers to Q. Following previous work (H. Sun, Dhingra, et al. 2018; H. Sun, Bedrax-Weiss, et al. 2019; H. Sun, Arnold, et al. 2020), we assume that the set of entities mentioned in the question $\mathcal{V}_Q \subseteq \mathcal{V}$ is given. These nodes are also called the *anchor nodes* of the question and in practice are commonly obtained using an entity-linking module.

Overview. KBQA can be cast as an entity seeking problem on $\mathcal{G}$ by translating Q into a set of nodes $\mathcal{V}_Q \subseteq \mathcal{V}$ (the starting points of the search) and seeking for nodes that provide an answer (H. Sun, Dhingra, et al. 2018; H. Sun, Bedrax-Weiss, et al. 2019; H. Sun, Arnold, et al. 2020). Attempting to find $\mathcal{A}_Q$ directly on $\mathcal{G}$ is prohibitive in practice, as even the most efficient graph-based neural networks generally scale at least linearly with the number of edges. Our approach mitigates this issue by breaking the problem in two subproblems.

(a) We first utilize a *relation-level* model ϕ to obtain a set of *candidate answers* $\tilde{\mathcal{A}}_Q$, such that $\mathcal{A}_Q \subseteq \tilde{\mathcal{A}}_Q$. We refer to ϕ as "*relation-level*" because, as we will see, it operates on the *coalesced graph*, a simplified representation of $\mathcal{G}$, where edges of the same relation type are coalesced. The coalesced graph is constructed before training and incurs a one-time linear cost. By exploiting it during training, the relation-level model scales with the number of (distinct) relation types in the KG, which are usually significantly fewer than the number of edges or entities.

(b) The candidate answers are then refined using an *edge-level* model ψ applied on a subgraph $\mathcal{G}(\tilde{\mathcal{A}}_Q)$ of the original knowledge graph in the vicinity of $\tilde{\mathcal{A}}_Q$. We should note that the refining step is not always necessary. Indeed, we found that a relation-level model is sufficient to perfectly solve tasks like multi-hop question answering (Zhang et al. 2018). Figure 5.1 shows an overview of our approach.

5.2.1 Relational coalescing for efficient knowledge seeking

Our approach relies on a relation-level model ϕ that operates as a knowledge seeker in $\mathcal{G}$. The model identifies a node v as a candidate $v \in \tilde{\mathcal{A}}_Q$ based on the sequence of relations that connect it with $\mathcal{V}_Q$. This can be achieved by using a neural network ϕ to predict how likely it is that the correct answer is reached from $\mathcal{V}_Q$ by following a sequence of relations R.

Reachability. To define how our method works, it will help to formalize the concept of reachability. Let $R = (r_1, \ldots, r_{|R|})$ be a sequence of relations. We say that "v *is* R*-reachable from* $\mathcal{V}_Q$" if there exists a path $P = (v_1, \ldots, v_{|R|}, v)$ in $\mathcal{G}$ such that:

$$v_1 \in \mathcal{V}_Q \quad \text{and} \quad v_i \xrightarrow{r_i} v_{i+1} \quad \text{for every} \quad i = 1, \ldots, |R|.$$

That is, we can reach v by starting from a node in $\mathcal{V}_Q$ and following a sequence of edges with relation types R. We also denote by *reach*$_\mathcal{G}(\mathcal{V}_Q, R)$ the set of nodes that are

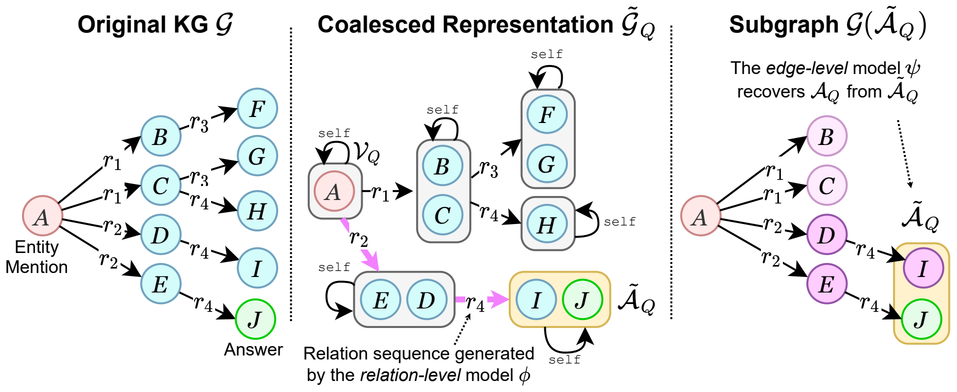

Original KG $\mathcal{G}$
Entity Mention
Answer
Coalesced Representation $\tilde{\mathcal{G}}_Q$
self
$\mathcal{V}_Q$
$\tilde{\mathcal{A}}_Q$
Relation sequence generated by the *relation-level* model ϕ
Subgraph $\mathcal{G}(\tilde{\mathcal{A}}_Q)$
The *edge-level* model ψ recovers $\mathcal{A}_Q$ from $\tilde{\mathcal{A}}_Q$

Figure 5.1: Overview of our approach. A *relation-level* model operates on a coalesced representation of the original KG to generate a set of candidate answers $\tilde{\mathcal{A}}_Q$. This approximate solution is then refined by an *edge-level* model applied on a subgraph of the original KG.

R-reachable from $\mathcal{V}_Q$:

$$reach_{\mathcal{G}}(\mathcal{V}_Q, R) = \{v \in \mathcal{V} \mid v \text{ is } R\text{-reachable from } \mathcal{V}_Q\}.$$

Relational coalescing. Given a knowledge graph $\mathcal{G}$, a question Q, and a set of entity mentions $\mathcal{V}_Q$, we consider a representation of the graph $\tilde{\mathcal{G}}_Q = (\tilde{\mathcal{V}}_Q, \tilde{\mathcal{R}}_Q, \tilde{\mathcal{E}}_Q)$, which allows us to efficiently compute sets of nodes that are reachable from $\mathcal{V}_Q$. We refer to this representation as the question-dependent coalesced KG, because edges with the same relation type are coalesced, as shown in Figure 5.1. The nodes of $\tilde{\mathcal{G}}_Q$ are sets of nodes of $\mathcal{G}$ that are reachable from $\mathcal{V}_Q$ by following any possible sequence of relations originating from $\mathcal{V}_Q$. The graph $\tilde{\mathcal{G}}_Q$ has an edge $\mathcal{V}_i \xrightarrow{r} \mathcal{V}_j$ if $\mathcal{V}_j$ is the set of nodes that are reachable from $\mathcal{V}_i$ by following relation r. For convenience, we include a relation type `self` $\in \tilde{\mathcal{R}}_Q$ to denote self loops. We refer the reader to Appendix C.1 for a formal definition of $\tilde{\mathcal{G}}_Q$. The coalesced graph can be precomputed once as a preprocessing step for each question Q and incurs a one-time linear cost. In practice, however, we do not need to compute and store all the nodes in $\tilde{\mathcal{G}}_Q$ but only edge labels. This makes learning efficient because each forward/backward pass scales with the number of relation types and does not depend on the number of nodes or edges in the KG.

Knowledge seeking in $\tilde{\mathcal{G}}_Q$. The coalesced graph allows us to provide approximate answers to input questions in an efficient manner. Specifically, we seek $k \geq 1$ sequences of relations $R_i^\star$, such that:

$$\mathcal{A}_Q \subseteq \tilde{\mathcal{A}}_Q = \bigcup_{i=1}^{k} reach_{\mathcal{G}}(\mathcal{V}_Q, R_i^\star).$$

We can achieve this by using a model ϕ that only considers relation sequences originating from $\mathcal{V}_Q$. The model predicts the likelihood $\phi : \tilde{\mathcal{E}}_Q \to [0,1]$ of following a certain edge in a relation sequence from $\mathcal{V}_Q$ to $\tilde{\mathcal{A}}_Q$. Then, given $R = (r_1, \dots, r_{|R|})$ and a node in the coalesced graph $\mathcal{V}_Q$, we can compute the likelihood of R by multiplying the likelihood of all edges traversed by R in $\tilde{\mathcal{G}}_Q$:

$$\mathsf{P}(R \mid Q, \tilde{\mathcal{G}}_Q, \mathcal{V}_Q) \propto \prod_{i=1}^{|R|} \phi(\mathit{reach}_{\mathcal{G}}(\mathcal{V}_Q, R_{1\to i-1}), r_i, \mathit{reach}_{\mathcal{G}}(\mathcal{V}_Q, R_{1\to i}) \mid Q),$$

where $R_{1\to i} = (r_1, \dots, r_i)$ is the subsequence of R up to the i-th relation. We generate $\tilde{\mathcal{A}}_Q$ by selecting the top k relation sequences $R_i^\star$ with maximum likelihood $\mathsf{P}(R_i^\star \mid Q, \tilde{\mathcal{G}}_Q, \mathcal{V}_Q)$. This can be done by an efficient search algorithm, such as beam search starting from $\mathcal{V}_Q$. Then, we compute $\tilde{\mathcal{A}}_Q$ as the union of all target nodes of the selected relation sequences. More details about the knowledge-seeking algorithm are provided in Appendix $C.2$.

5.2.2 Refining the solution on the original KG

In certain cases, like multi-hop question answering (Zhang et al. 2018), the set of candidate answers $\tilde{\mathcal{A}}_Q$ may already be a reasonable estimate of $\mathcal{A}_Q$. We will substantiate this claim experimentally in Section 5.4. In general, however, we recover $\mathcal{A}_Q$ by using an *edge-level* model ψ applied on a subgraph $\mathcal{G}(\tilde{\mathcal{A}}_Q)$ of $\mathcal{G}$. Specifically, we construct $\mathcal{G}(\tilde{\mathcal{A}}_Q)$ as the subgraph induced by the set of nodes $\mathcal{V}(\tilde{\mathcal{A}}_Q)$, which includes all nodes visited when following the top-k relation sequences along with their neighbors (see Figure 5.1 for an example). Any existing method for KBQA can be used to instantiate ψ by running it on $\mathcal{G}(\tilde{\mathcal{A}}_Q)$ rather than $\mathcal{G}$. We opted to use a Graph Convolutional Network (GCN) conditioned on the input question with the same architecture as in (H. Sun, Dhingra, et al. 2018). The edge-level model is constrained to predict an answer among the candidates generated by the relation-level model.

5.2.3 Analysis of scalability and expressive power

This section provides a scalability analysis of our approach and shows that the relation-level model scales linearly with the number of relation types in the graph. Then, we analyse the expressive power of SQALER and we show the class of logical queries that it can answer.

Computational complexity. As mentioned, we do not evaluate the likelihood ϕ for all edges in $\tilde{\mathcal{G}}_Q$, but we generate the most likely relation sequences using a knowledge-seeking procedure based on the beam search algorithm. At any given time step, only the β most likely relation sequences are retained and further explored at the next iteration. Hence, the time complexity required by our algorithm is $\mathcal{O}(\tau_{max} \cdot \beta \cdot d^+_{max}(\tilde{\mathcal{G}}_Q))$, where τ_{max} is the maximum allowed number of decoding time steps and $d^+_{max}(\tilde{\mathcal{G}}_Q)$ is the maximum outdegree of $\tilde{\mathcal{G}}_Q$. Note that $d^+_{max}(\tilde{\mathcal{G}}_Q)$ is bounded by the number of relations in the graph, whereas τ_{max} and β are constant parameters of the algorithm and are usually small. This gives a time complexity of:

$$\mathcal{O}(\tau_{max} \cdot \beta \cdot |\mathcal{R}|) = \mathcal{O}(|\mathcal{R}|).$$

Hence, the knowledge-seeking algorithm scales linearly with the number of relations in the KG. The space complexity is also $\mathcal{O}(\tau_{max} \cdot \beta \cdot |\mathcal{R}|)$. A more detailed analysis is provided in Appendix C.2.

Expressive power. Given a natural language question Q, we can represent the inferential chain needed to obtain $\mathcal{A}_Q$ from $\mathcal{V}_Q$ as a logical query Q on $\mathcal{G}$. As an example, the question in Figure 5.2, *"Who starred in films directed by George Lucas?"*, can be represented by the logical query: $\mathsf{Q}[\mathsf{V}_?] = \mathsf{V}_?.\exists \mathsf{V} : \mathsf{Directed}(\mathsf{George\_Lucas}, \mathsf{V}) \wedge \mathsf{Starred}(\mathsf{V}, \mathsf{V}_?)$. We denote with $\mathsf{V}_?$ the target variable of the query and we say that $v \in \mathcal{V}$ satisfies Q if $\mathsf{Q}[v] = \mathsf{True}$. A query Q is an *existential positive first-order (EPFO) query* if it involves the existential quantification ($\exists$), conjunction ($\wedge$), and disjunction ($\vee$) (Dalvi et al. 2012) of literals corresponding to relations in the KG. Each literal is of the form $r(\mathsf{V}, \mathsf{V}')$, where V is either a node in $\mathcal{V}_Q$ or an existentially quantified bound variable, and V' is either an existentially quantified bound variable or the target variable. A literal $r(\mathsf{V}, \mathsf{V}')$ is satisfied if $\mathsf{V} \xrightarrow{r} \mathsf{V}'$, for $r \in \mathcal{R}$. Any EPFO query can be represented in *disjunctive normal form* (DNF) (Davey et al. 2002), namely as a disjunction of conjunctions. Note that, we do not consider queries with universal quantification ($\forall$), as we assume that in real-world KGs no entity connects to all the others. Then, the following theorem holds for any knowledge graph and EPFO query.

Theorem 5.2.1. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ be a knowledge graph and $\mathcal{V}_Q \subseteq \mathcal{V}$ denote a set of entities in $\mathcal{G}$. Let Q be a valid existential positive first-order query on $\mathcal{G}$ and let $n_\vee$ be the number of disjunction operators in the disjunctive normal form of Q. Then, there exist $k \leq n_\vee + 1$ sequences of relations $R_i^\star \in \mathcal{R}^*$ such that:*

$$\mathcal{A}_Q \subseteq \bigcup_{i=1}^{k} \mathit{reach}_{\mathcal{G}}(\mathcal{V}_Q, R_i^\star),$$

where $\mathcal{A}_Q = \{v \in \mathcal{V} \mid \mathsf{Q}[v] = \mathsf{True}\}$ is the denotation set of Q, namely the entities satisfying Q.

This shows that sampling $n_\vee + 1$ sequences of relations allows generating a set of candidate answers $\tilde{\mathcal{A}}_Q$ that does not miss any of the real answers $\mathcal{A}_Q$. Then, assuming that the edge-level model ψ can recover $\mathcal{A}_Q$ from $\tilde{\mathcal{A}}_Q$, our approach can be used to answer any EPFO query on $\mathcal{G}$. More details about the expressive power of SQALER and the proof of Theorem 5.2.1 are provided in Appendix C.3.

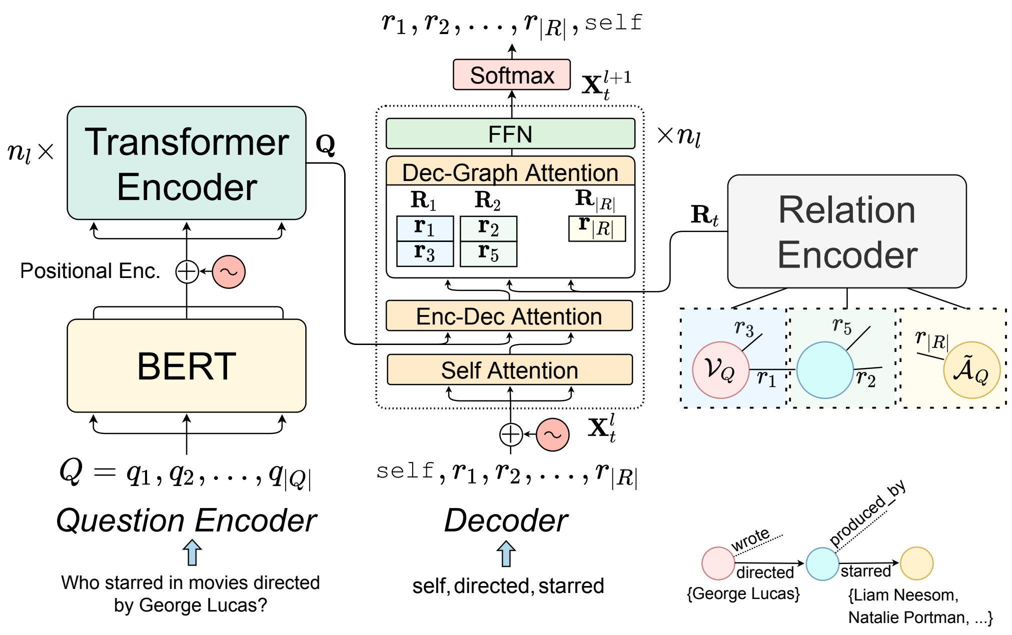

$r_1, r_2, \ldots, r_{|R|}$, self
Softmax
$\mathbf{X}_t^{l+1}$
$n_l \times$
Transformer Encoder
$\mathbf{Q}$
FFN
$\times n_l$
Dec-Graph Attention
$\mathbf{R}_1$ $\mathbf{R}_2$ $\mathbf{R}_{|R|}$
$\mathbf{r}_1$ $\mathbf{r}_3$ $\mathbf{r}_2$ $\mathbf{r}_5$ $\mathbf{r}_{|R|}$
$\mathbf{R}_t$
Relation Encoder
Positional Enc.
Enc-Dec Attention
r_3 r_5 $r_{|R|}$
$\mathcal{V}_Q$ r_1 r_2 $\tilde{\mathcal{A}}_Q$
BERT
Self Attention
$\mathbf{X}_t^l$
$Q = q_1, q_2, \ldots, q_{|Q|}$
self, $r_1, r_2, \ldots, r_{|R|}$
Question Encoder
Decoder
Who starred in movies directed by George Lucas?
self, directed, starred
wrote
produced_by
directed
starred
{George Lucas}
{Liam Neesom, Natalie Portman, ...}

Figure 5.2: Architecture of the SQALER relation-level model. A question encoder is used to obtain a representation of tokens in the input natural language question. Then a graph-guided decoder is applied to obtain the likelihood of output relation sequences. The decoder is constrained to only attend to valid relations according to the structure of the coalesced KG.

5.3 Architecture of the relation-level model

For the relation-level model ϕ, we propose an auto-encoder, where the the decoder is constrained to follow sequences of relations in the coalesced representation $\tilde{\mathcal{G}}_Q$. We train the network with weak supervision, assuming that a sequence of relations is correct if it reaches a set of candidate answers $\tilde{\mathcal{A}}_Q$ that is the smallest reachable superset of the $\mathcal{A}_Q$. We found it useful to *pretrain* the model in order to infuse knowledge from the KG. In this case, we train the model to predict a path in the KG, given the representations of the source and target nodes. More details about training strategies are given in Appendix C.4.

The architecture of the model, as shown in Figure 5.2, includes three main components: a *question encoder*, a *relation encoder* and a *graph-guided decoder*. We explain each one below.

Question encoder. The encoder receives as input a natural language question, which comprises a sequence of tokens $Q = (q_1, q_2, \ldots, q_{|Q|})$. The question is encoded using a pre-trained BERT (Devlin et al. 2019) model and processed with the same positional encoding technique used in (Vaswani et al. 2017). The resulting embeddings are then fed into $n_l = 3$ transformer encoder layers (Vaswani et al. 2017). This results in a matrix $\mathbf{Q} \in \mathbb{R}^{|Q|+1 \times d_{model}}$, where the first row vector is an overall representation of the whole query Q (derived from the embedding of the `[CLS]` token introduced by BERT) and each remaining row represents the final d_{model}-dimensional encoding of a token in the input question.

Relation encoder. The relation encoder produces a representation $\mathbf{r} \in \mathbb{R}^{d_{model}}$ for each relation type $r \in \mathcal{R}$. We decided to encode relations based on their surface form, with the same pre-trained BERT model used in the question encoder. In this case, only the embedding of the `[CLS]` token is used in order to get the final representation $\mathbf{r}$ of each relation type $r \in \mathcal{R}$. At inference time, or in case the BERT model is not fine-tuned, the embeddings of the relations can be precomputed as a preprocessing step to improve the efficiency of the approach.

Graph-guided decoder. The decoder's job is to predict a sequence of relations leading from $\mathcal{V}_Q$ to $\tilde{\mathcal{A}}_Q$ in $\tilde{\mathcal{G}}_Q$. At any time step t, it receives as input a sequence of relations $R_t = (\texttt{self}, r_1, \ldots, r_{t-1})$ and predicts the next relation r_t (`self` is used as a special token to denote the start of decoding). Note that the input sequence uniquely determines a node $\mathcal{V}_t$ in the graph $\tilde{\mathcal{G}}_Q$, namely the node reachable from $\mathcal{V}_Q$ by following R_t. The decoder thus selects r_t by choosing amongst the outgoing edges $\tilde{\mathcal{E}}_t$ of $\mathcal{V}_t$. We use the same number of layers n_l both for the question encoder and the decoder. Let $\mathbf{X}_t^l = [\mathbf{x}_0^l, \ldots, \mathbf{x}_{t-1}^l]^\top \in \mathbb{R}^{t \times d_{model}}$ denote the hidden state of the l-th layer of the decoder preceding time step t. Note that $\mathbf{X}_t^0$ is the representation of the sequence R_t, obtained by using the relation encoder described above and the same positional encoding technique used in the question encoder. For each decoder layer, we perform self-attention over the target sequence $\mathbf{X}_t^l$ by computing:

$$\bar{\mathbf{x}}_t^l = \mathit{Attention}(\mathbf{x}_t^l, \mathbf{X}_t^l, \mathbf{X}_t^l),$$

where *Attention* is a function that performs multi-head scaled dot-product attention (Vaswani et al. 2017) with skip connections and layer normalization (Ba et al. 2016). The above step allows each relation in the decoded sequence to attend to all the others predicted up to time step t. We then let the result attend to the question as:

$$\bar{\mathbf{x}}_t^{Q,l} = \mathit{Attention}(\bar{\mathbf{x}}_t^l, \mathbf{Q}, \mathbf{Q}).$$

This is done in order to update the current state of the decoder based on the input question. Next, let $\mathbf{R}_t \in \mathbb{R}^{|\tilde{\mathcal{E}}_t| \times d_{model}}$ denote the encoding of the relations labeling all edges in $\tilde{\mathcal{E}}_t$. We constrain the decoded sequence to follow the structure of the graph by attending only to valid relations as follows:

$$\bar{\mathbf{x}}_t^{R,l} = \mathit{Attention}(\bar{\mathbf{x}}_t^{Q,l}, \mathbf{R}_t, \mathbf{R}_t).$$

We get the hidden state of the next layer $\mathbf{x}_t^{l+1}$ by processing the result with a feed forward network. The model outputs a categorical distribution $\phi(e \mid Q) \in [0, 1]$ over the edges $e \in \tilde{\mathcal{E}}_t$, by applying a softmax function as follows:

$$\phi(\mathcal{V}_i \xrightarrow{r} \mathcal{V}_j \mid Q) = \frac{\exp(\mathbf{r}^\top \mathbf{x}_t^{n_l})}{\sum_{\mathcal{V}'_i \xrightarrow{r'} \mathcal{V}'_j \in \tilde{\mathcal{E}}_t} \exp(\mathbf{r'}^\top \mathbf{x}_t^{n_l})},$$

where $\mathbf{x}_t^{n_l}$ is the output of the final layer of the decoder, whereas $\mathbf{r}$ and $\mathbf{r}'$ denote the representations of relations r and r' respectively.

5.4 Experiments

This section presents an evaluation of our approach with respect to both reasoning performance and scalability. We first show that **SQALER** reaches state-of-the-art results on popular KBQA benchmarks and can generalize compositionally out of the training distribution. Then, we demonstrate the scalability of our approach on KGs with millions of nodes. We refer the reader to Appendix C.5 for more details.

5.4.1 Experimental setup

Datasets. We evaluate the reasoning performance of our approach on *MetaQA* (Zhang et al. 2018) and *WebQuestionsSP* (Yih et al. 2015). *MetaQA* includes multi-hop questions over the WikiMovies KB (Miller et al. 2016) and we consider both 2-hop (**MetaQA 2**) and 3-hop (**MetaQA 3**) queries. *WebQuestionsSP* (**WebQSP**) comprises more complex questions answerable over a subset of Freebase (Bollacker et al. 2008), a large KG with millions of entities. We further assess the compositional generalization ability of **SQALER** on the *Compositional Freebase Questions* (*CFQ*) dataset (Keysers et al. 2020). Each question in *CFQ* is obtained by composing primitive elements (*atoms*). Whereas the training and test distribution of atoms are similar, the test set contains different *compounds*, namely new ways of composing these atoms. *CFQ* comprises three dataset splits (**MCD1**, **MCD2**, and **MCD3**), with maximal compound divergence (MCD) between the training and test distributions. We refer the reader to Appendix C.5.1 for an extensive description of the datasets.

Evaluation protocol. In our experiments on *MetaQA* and *WebQuestionsSP*, we assess the performance of three variants of our approach: (a) a version that only makes use of the relation-level model without the refinement step (**SQALER – Unrefined**), (b) a model that utilizes a key-value memory network to identify the correct answers from the candidates (**SQALER – KV-MemNN**), and (c) a model that uses a GNN architecture for the refinement step (**SQALER – GNN**), as explained in Section 5.2.2. Following previous work (H. Sun, Dhingra, et al. 2018; H. Sun, Bedrax-Weiss, et al. 2019; H. Sun, Arnold, et al. 2020; Saxena et al. 2020), we evaluate the models based on the *Hits@1* metric. On the *CFQ* dataset, we evaluate the accuracy of the refined model with the

	MetaQA 2	MetaQA 3	WebQSP
KV-MemNN (Miller et al. 2016)	82.7	48.9	46.7
GRAFT-Net (H. Sun, Dhingra, et al. 2018)	94.8	77.7	70.3
ReifKB + mask (Cohen, H. Sun, et al. 2020)	95.4	79.7	52.7
PullNet (H. Sun, Bedrax-Weiss, et al. 2019)	99.9	91.4	69.7
EmbedKGQA (Saxena et al. 2020)	98.8	94.8	66.6
EmQL (H. Sun, Arnold, et al. 2020)	98.6	99.1	75.5
SQALER – Unrefined	99.9	99.9	70.6
SQALER – KV-MemNN	99.9	99.9	72.1
SQALER – GNN	99.9	99.9	76.1

Table 5.1: Hits@1 on *MetaQA* and *WebQuestionsSP*

GNN based on whether it predicts exactly the same answers given by the corresponding SPARQL query.

5.4.2 Main results

KBQA Performance. Table 5.1 summarizes the results of our experiments on the two benchmark datasets. For the two multi-hop *MetaQA* datasets, we achieve state-of-the-art performance by only using the relation-level model of **SQALER**. As shown in Table 5.1, **SQALER** outperforms all the baselines on **MetaQA 3**, demonstrating the ability of our approach to perform multi-hop reasoning over a KG. For the more complex questions in the *WebQuestionsSP* dataset, the unrefined **SQALER** model achieves better performance than all but one (**EmQL**) of the baselines. To achieve such performance, however, EmQL creates a custom set of logical operations tailored towards the specifics of the target KG and the kind of questions in the dataset, while our approach is agnostic with respect to such details. Combining the relation and edge-level models improves the performance on **WebQSP**. In particular, **SQALER – GNN** outperforms all considered baselines on the three datasets.

Compositional generalization. In order to evaluate the compositional generalization ability of **SQALER**, we performed additional experiments on the *CFQ* dataset. Table 5.2 shows the accuracy on the three MCD splits and the mean accuracy (**MCD-mean**) in comparison to the other methods in the leaderboard. Note that the other approaches address a semantic parsing task and require additional supervision, as they are trained to predict the target query. On the other hand, we aim to predict directly the set of answers to the input question. The experiment shows that **SQALER** is able to achieve compositional generalization with an accuracy comparable to the state-of-the-art model on *CFQ* for semantic parsing.

Subgraph extraction. We analyzed the candidate solutions produced by the relation-level model in order to evaluate the suitability of our approach to building small

	MCD1	MCD2	MCD3	MCD-mean
LSTM + Attention (Keysers et al. 2020)	0.289 ± 0.018	0.050 ± 0.008	0.108 ± 0.006	0.149 ± 0.011
Transformer (Vaswani et al. 2017)	0.349 ± 0.011	0.082 ± 0.003	0.106 ± 0.011	0.179 ± 0.009
Universal Transformer (Dehghani et al. 2019)	0.374 ± 0.022	0.081 ± 0.016	0.113 ± 0.003	0.189 ± 0.014
Evolved Transformer (So et al. 2019)	0.424 ± 0.010	0.093 ± 0.008	0.108 ± 0.002	0.208 ± 0.007
T5-11B (Raffel et al. 2020)	0.614 ± 0.048	0.301 ± 0.022	0.312 ± 0.057	0.409 ± 0.043
T5-11B-mod (J. Guo et al. 2019)	0.616 ± 0.124	0.313 ± 0.128	0.333 ± 0.023	0.421 ± 0.091
HPD (Y. Guo et al. 2020)	0.720 ± 0.075	0.661 ± 0.064	0.639 ± 0.057	0.673 ± 0.041
SQALER – GNN	0.734 ± 0.039	0.653 ± 0.040	0.627 ± 0.045	0.671 ± 0.041

Table 5.2: Accuracy and 95% confidence interval on the *CFQ* dataset

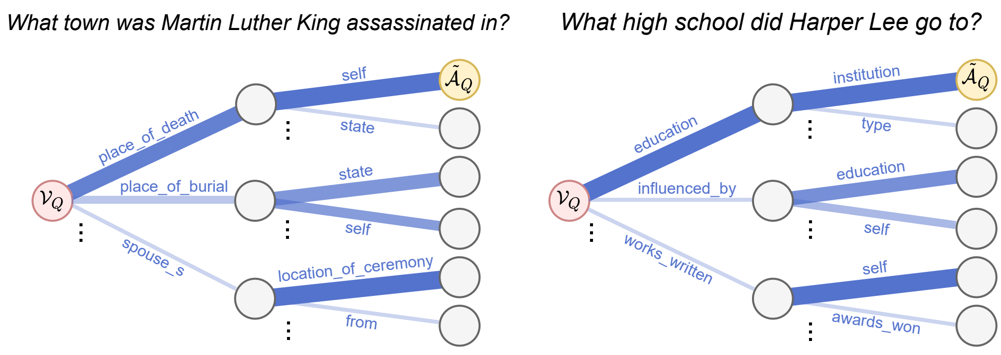


Figure 5.3: Attention weights given by the relation-level model to the edges of the coalesced graph for two questions in *WebQuestionsSP*. Thicker and darker edges represent higher attention weights.

question subgraphs that are likely to contain the answers to a natural language question. For this purpose, we computed the precision and recall of the set of candidate answers with varying number of relation sequences sampled by the relation-level model. Figure 5.3 shows the top relation sequences predicted by the relation-level model on two questions from the test set of *WebQuestionsSP*. The precision and recall curves are shown in Figure 5.4. As expected, on *MetaQA* the recall is high for all values of k, because selecting the most likely sequence of relations is sufficient to solve the multi-hop question answering task. On *WebQuestionsSP*, only 3 sequences of relations are sufficient to obtain a recall of 0.91, and we can improve it to 0.95 by generating still small subgraphs consisting of only 10 sequences of relations.

5.4.3 Efficiency and scalability

We analyze the efficiency of our approach on synthetic KBs (as in (Cohen, F. Yang, et al. 2017; Cohen, H. Sun, et al. 2020)) and then compare the scalability of different preprocessing methods on the KGs of *MetaQA* and *WebQuestiontsSP*. First, we perform experiments on KBs where the relational coalescing has no effect: the outdegree of each node is equal to the number of relation types and all edges originating from a node have different relation labels. We perform two experiments on such KBs. In

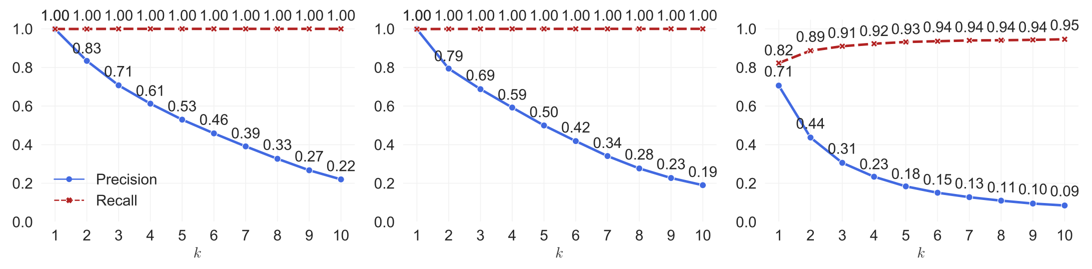


Figure 5.4: Precision and recall of the top k sequences of relations on MetaQA 2 (left), MetaQA 3 (center) and WebQSP (right)

the first one (Figure 5.5a), the number of relation types is fixed to $|\mathcal{R}| = 10$ and the number of entities varies from $|\mathcal{V}| = 10^2$ to $|\mathcal{V}| = 10^6$. In the second task (Figure 5.5b), the number of entities is fixed to $|\mathcal{V}| = 5000$ and the number of relations varies from $|\mathcal{R}| = 1$ to $|\mathcal{R}| = 10^3$. The single answer node is always two-hops away from the entities mentioned in the question. We compare **SQALER** (unrefined) against a GNN-based approach (**GRAFT-Net** (H. Sun, Dhingra, et al. 2018)) and a key-value memory network (**KV-MemNN** (Miller et al. 2016)). The approaches are evaluated based on the queries per second at inference time with a mini-batch size of 1. The results show that increasing the number of entities has negligible impact on the performance of **SQALER**, whereas GRAFT-Net and the key-value memory network are limited to graphs with less than 10k nodes. This shows that, in large KGs like Freebase, the baselines would not be able to handle even a 2-hop neighborhood of the entities mentioned in the question (we refer the reader to Appendix C.5.5 for more details). Finally, from the results in Figure 5.5b, we see that the throughput of our approach decreases with the number of relation types. However, in practice, we can leverage the GPU to score the edges of the graph in parallel. This is why we observe only a minor drop in performance when the number of relation types grows from $|\mathcal{R}| = 1$ to $|\mathcal{R}| = 100$.

In order to assess the scalability of the proposed relational coalescing operation, we further compare commonly used preprocessing methods on the KG of *WebQuestionsSP*. We evaluate the time required to extract complete 2-hop neighborhoods of the entities mentioned in the question and the time to perform Personalized Page Rank (PPR) on such graphs. The results are shown in Figure 5.5c. Note that, at inference time, we can perform the coalescing only on the portion of the graph explored by the model, which makes **SQALER** much more efficient. At training time, the preprocessing is comparable to the 2-hop neighborhood extraction. Finally, Figure 5.5d shows the performance of the models with the respective preprocessing step at inference time on synthetic KBs with growing number of edges.

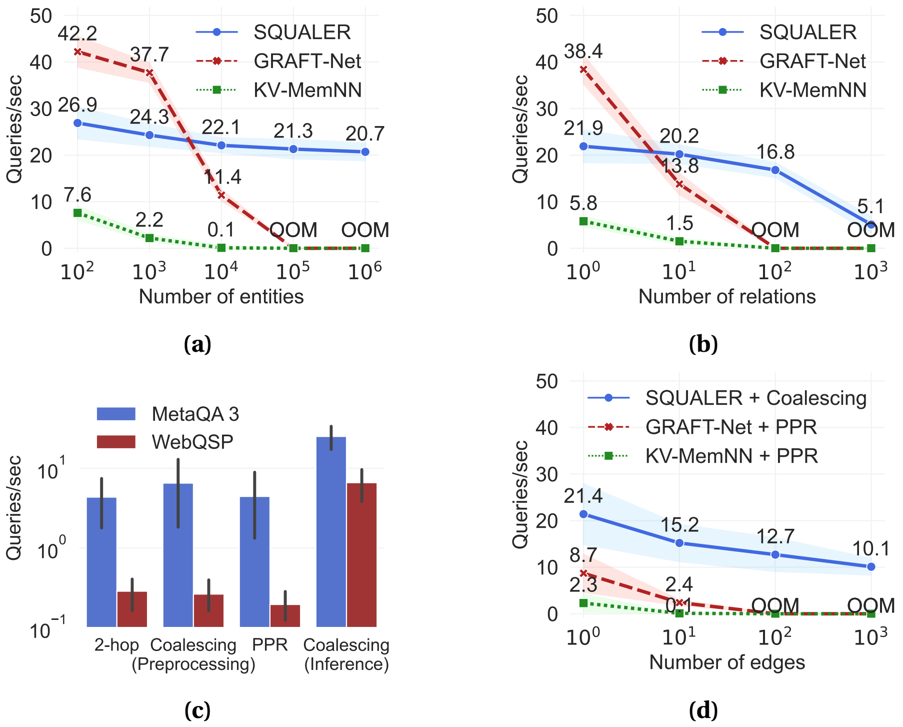


Figure 5.5: Inference time in queries/sec on synthetic KBs with increasing number of entities (a) and relation types (b). Time required by different preprocessing steps on the KG of *WebQuestionsSP* and *MetaQA* (c). Complete inference and preprocessing time on synthetic KBs with increasing number of edges (d). We set the queries/sec to 0 when the model runs out-of-memory (OOM).

5.4.4 Incomplete knowledge graphs

In order to evaluate the capability of our approach to cope with missing information in the knowledge graph, we performed two additional experiments. In the first experiment, we evaluated our approach (the **SQALER – GNN** variant) on *WebQuestionsSP* using incomplete knowledge graphs with only 50% of the original edges (**50% KG**). Then, following previous work (H. Sun, Dhingra, et al. 2018; H. Sun, Bedrax-Weiss, et al. 2019), we tried to mitigate the missing information using additional sources of external knowledge. In particular, for each question, we used the same text documents extracted from Wikipedia as done by H. Sun, Dhingra, et al. (2018) (**50% KG + Text**). In this experiment, the relation-level model is unaware of the additional source of knowledge, but the information from the text documents is infused into the edge-level GNN with the same strategy used in GRAFT-Net (H. Sun, Dhingra, et al. 2018) (note that this makes the edge-level GNN-based model essentially equivalent to the full version of GRAFT-Net, with both KG and text support). We compare our approach against **GRAFT-Net** and **PullNet**, namely the two baselines designed for open-domain question answering with incomplete KGs and text documents.

	50% KG	**50% KG + Text**
GRAFT-Net (H. Sun, Dhingra, et al. 2018)	48.2	49.9
PullNet (H. Sun, Bedrax-Weiss, et al. 2019)	50.3	51.9
SQALER – GNN	53.5	55.2

Table 5.3: Hits@1 on *WebQuestionsSP* with incomplete KGs (50% of the edges) and additional text documents

The results of the experiments are reported in Table 5.3. We observe that, despite not being designed for incomplete KGs, **SQALER** outperforms the baselines on both experimental settings. This is not surprising, as **GRAFT-Net** relies on a simple heuristic process to construct question subgraphs and **PullNet** is constrained to follow the structure of the incomplete graph, because its iterative retrieval process can only expand nodes that are reachable from the set of anchor entities. This means that, in principle, any node retrieved by **PullNet**'s iterative process can also be reached by **SQALER**'s relation-level model. Similarly to the baselines, we note only a minor gain in performance when using the text documents as an additional source of information.

5.5 Related work

Several lines of research in the past few years have focused on introducing deep learning approaches aimed at reasoning over structured knowledge. In particular, this chapter is closely related to methods for learning to traverse KGs (Das, S. Dhuliawala, et al. 2018; Das, Neelakantan, et al. 2017; Guu et al. 2015) and recent works on answering conjunctive queries using deep learning approaches (Hamilton et al. 2018; Daza et al. 2020). In this context, several KB and query embedding methods have been proposed (Q. Wang et al. 2017). Many KB embedding approaches support the same operation performed by our relation-level model, namely relation projection (Cohen, H. Sun, et al. 2020; H. Sun, Arnold, et al. 2020; Hamilton et al. 2018; Ren, Hu, et al. 2020). Some KB embedding methods also explicitly learn to follow chains of relations and traverse KGs (Guu et al. 2015; Y. Lin et al. 2015; Das, Neelakantan, et al. 2017). Notably, Query2Box (Ren, Hu, et al. 2020) is a query embedding method that represents sets using box embeddings and the more recent beta embeddings (Ren and Leskovec 2020) extend the framework to support a complete set of first-order logic operators. The main difference with our model is that these methods operate on vector space, whereas our approach is constrained on the graph structure and learns to traverse the KG while keeping the ability to scale to large graphs. Also, our method answers questions in natural language, while the above methods are primarily designed for query answering. Recently, H. Sun, Arnold, et al. (2020) introduced EmQL, a query embedding method which has also been integrated in a question answering model.

Other lines of research on KBQA have focused on unsupervised semantic parsing

(Atzeni and Atzori 2018c; Atzeni and Atzori 2018b; Atzeni and Atzori 2018a) or on the introduction of supervised models, like graph neural networks (GNNs) designed for reasoning over knowledge graphs (H. Sun, Dhingra, et al. 2018; H. Sun, Bedrax-Weiss, et al. 2019; Yasunaga et al. 2021). These approaches pose the KBQA problem as a node classification task. For this reason, they have been applied succesfully only on small query-dependent graphs. Cohen, H. Sun, et al. (2020) addressed the problem of creating a representation of a symbolic KB that enables building neural KB inference modules that are scalable enough to perform non-trivial inferences with large graphs. Another recent work (Saxena et al. 2020) has explored using KG embeddings for question answering and handle incompleteness in the KG.

In our work, we combine relation projection with an edge-level GNN to address the KBQA problem. The same idea of combining GNNs with relational following was introduced in Gretel (J. Cordonnier et al. 2019), which learns to complete natural paths in a graph given a path prefix. Also, our idea of accelerating GNNs by operating on a reduced graph representation has strong connections with graph coarsening and sparsification (Loukas 2019; Loukas and Vandergheynst 2018; Batson et al. 2013).

Methods based on reinforcement learning (RL) have also been proposed to perform multi-hop reasoning over knowledge graphs. Xiong et al. (2017) proposed DeepPath, which relies on a policy-based agent that learns to reason over multi-hop paths by sampling relations at each step. Also, Das, S. Dhuliawala, et al. (2018) introduced MINERVA, a RL agent that learns how to navigate the graph conditioned only on an input entity and on a query. These approaches are designed for simple query answering and KB completion rather than KBQA. A main difference with our work is that **SQALER** samples multiple paths and employs an edge-level model to reach higher expressivity.

5.6 Conclusion

This chapter introduced **SQALER**, a scalable approach to reasoning and question answering over KGs. Our method is expressive and can reach state-of-the-art performance on widely used and challenging datasets. Further, **SQALER** scales with the number of (distinct) relation types in the graph and can effectively handle large-scale knowledge graphs with millions of entities. Our empirical evaluation also showed that our approach can generalize compositionally and that it can be used to generate question-dependent subgraphs that strike a good trade-off between precision and recall. This may effectively boost the performance of future KBQA methods relying on our subgraph extraction as a preprocessing step.

Overall, our work proposes an improvement to existing KBQA technology which carries impact to several practical applications. Nevertheless, we remind that the deployment of such models needs to be done cautiously. KBQA replaces a mature technology (traditional KBs and query languages) with less understood methods. The underlying KB may be incomplete, contain misinformation or biases that could negatively affect the decisions of the learned model. We hope that our work will spur further research in this area and contribute to the development of reliable KBQA systems.

6 Infusing structured knowledge in downstream entity-linking tasks

6.1 Introduction

State-of-the-art approaches to entity linking, namely the task of linking mentions of entities in a text to the corresponding entries in a knowledge base (Ferragina et al. 2010; Ganea et al. 2017), are nowadays large *generative models* (Aghajanyan et al. 2022; De Cao et al. 2021) which perform entity retrieval in a autoregressive way, effectively capturing relations between the context of a mention and entity descriptions. Though this category of methods achieves the best results, the preferred choice in large-scale applications are often still methods based on *dense retrieval* (Botha et al. 2020; Ayoola et al. 2022; Wu et al. 2020), as they are easier to train and can be more than one order of magnitude faster (Ayoola et al. 2022). These approaches learn to represent entities and mentions separately in the same embedding space, so that at inference time, the method only requires encoding the mention and retrieving the most similar entity. Despite being efficient, methods based on dense retrieval have the drawback of being very sensitive to the structure of the embedding space, thereby reaching lower accuracy compared to generative models.

To address this issue, in this chapter we aim to close the gap with generative approaches by infusing structural information in the latent space of retrieval-based methods. Recent work (Mulang et al. 2020; Ayoola et al. 2022) has shown the benefit of infusing prior factual knowledge in the models. In particular, Raiman et al. (2018) reported that prior knowledge of the type of a mention would result in nearly perfect disambiguation performance (99.0 micro-F_1 points on the popular AIDA-CoNLL benchmark of Hoffart et al. (2011)). Motivated by this insight, prior methods used type labels extracted from knowledge graphs (KGs) to improve downstream results (Ayoola et al. 2022; S. Chen et al. 2020; Orr et al. 2021), but as KGs can be highly incomplete, we aim to define type information in a fuzzy and more fine-grained manner.

Inspired by the concept of *duck typing* in programming languages, in this chap-

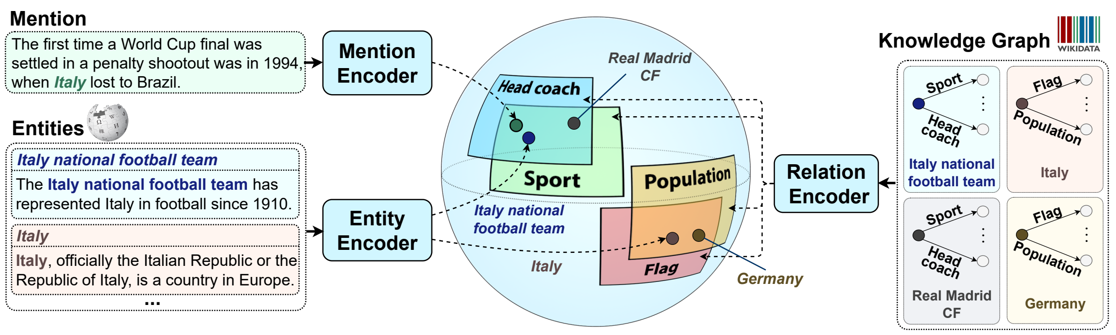

Mention
The first time a World Cup final was settled in a penalty shootout was in 1994, when Italy lost to Brazil.
Mention Encoder
Entities
Italy national football team
The Italy national football team has represented Italy in football since 1910.
Italy
Italy, officially the Italian Republic or the Republic of Italy, is a country in Europe.
...
Entity Encoder
Head coach
Real Madrid CF
Sport
Population
Italy national football team
Italy
Flag
Germany
Relation Encoder
Knowledge Graph
WIKIDATA
Sport
Head coach
Italy national football team
Flag
Population
Italy
Sport
Head coach
Real Madrid CF
Flag
Population
Germany

Figure 6.1: Entity disambiguation flow in DUCK. A mention encoder and an entity encoder learn to represent mentions and entity descriptions respectively. Following the concept of *duck typing*, relations in a knowledge graph are used to determine entity types (the right side of the figure shows how we can distinguish football teams and countries based on their relations). Relations are represented as box embeddings in polar coordinates and the model is optimized to place entities inside the boxes corresponding to their relations.

ter we propose DUCK (*Disambiguating Using Categories extracted from Knowledge*), a novel approach to infusing prior type information in the latent space of entity-disambiguation methods based on dense retrieval. Extending *duck typing* to the realm of knowledge graphs, we loosely define the type of an entity based on the relations that it has with other entities in the graph. An example showing how the relations in a KG like Wikidata (Vrandečić et al. 2014) can be used to determine the type of an entity is depicted in Figure 6.1 (right).

Building on recent work on region-based representations (Vilnis et al. 2018; Dasgupta et al. 2020; Abboud et al. 2020), we introduce box embeddings in spherical polar coordinates and we propose to model relations using this representation, as shown in Figure 6.1. Then, we optimize the model to structure the latent space in such a way that entities fall within the boxes corresponding to their relations, so that entities that share many relations (which are assumed to be of the same type in our duck-typing formulation) will be clustered together.

We used our approach to train a bi-encoder model with the same architecture of Wu et al. (2020) and we performed a thorough evaluation of DUCK on popular entity disambiguation benchmarks. Our experiments show that DUCK achieves new state-of-the-art results on well-known datasets, exceeds the performance of other type-aware models (trained on 10 times more data), and matches the overall results of much more expensive generative models, with more than 18 times more parameters than DUCK. Moreover, our ablation studies show that incorporating type information using box embeddings in polar coordinates improves the performance of the model by up to 7.9 micro-F_1 points. Finally, qualitative analyses provide evidence that our definition of *duck typing* results in a clear clustering of types and that our model is able to predict the correct relations of an entity despite the incompleteness of the KG.

6.2 Preliminaries

To start with, we briefly introduce some relevant background that our work builds on and we formalize the entity-disambiguation problem.

Problem statement. The goal of *entity disambiguation* (ED) is to link entity mentions in a piece of text to the entity they refer to in a reference knowledge base (KB). For each entity e, we assume we have an entity description expressed as a sequence of tokens $s_e = (s_e^{(1)}, \ldots, s_e^{(|s_e|)})$. Similarly, each mention m is associated with a sequence of tokens $s_m = (s_m^{(1)}, \ldots, s_m^{(|m|)})$, representing the mention itself and its context. We further assume that the reference KB is a knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ is a set of entities and $\mathcal{R}$ is a set of relations, namely boolean functions $r : \mathcal{E} \times \mathcal{E} \to \{0, 1\}$ denoting whether a relation exists between two entities in $\mathcal{E}$. Then, given a set of entity-mention pairs $\mathcal{D} = \{(m_1, e^\star_{m_1}), \ldots, (m_{|\mathcal{D}|}, e^\star_{m_{|\mathcal{D}|}})\}$, we aim to learn a model $f : \mathcal{M} \to \mathcal{E}$, such that the entity predicted by the model for a given mention $\hat{e}_m = f(m)$ is the correct entity $e^\star_m$. Notice that, in this chapter, we use a slightly different notation for knowledge graphs and entities. This is done for ease of notation and to be more aligned with the literature on entity linking.

Dense-retrieval methods. Methods based on dense retrieval (Wu et al. 2020; Ayoola et al. 2022) learn to represent mentions and entities in the same latent space, often optimizing a cross-entropy objective of the form:

$$\mathcal{L}_{ED}(m) = -s(m, e^\star_m) + \log \sum_j \exp(s(m, e_j)),$$

where s is a similarity function between entities and mentions. This objective encourages the representation of mention m to be close to the representation of the correct entity $e^\star_m$ and far from other entities e_j. The similarity function $s(m, e)$ between a mention m and an entity e is usually chosen to be the dot product between learned representations $\boldsymbol{m}, \boldsymbol{e} \in \mathbb{R}^d$ of the mention and entity respectively.

6.3 Duck: enhancing entity disambiguation with duck typing

Our method builds on dense-retrieval methods and aims to enhance their performance using fine-grained type information.

6.3.1 Modeling fine-grained type information from knowledge graphs

Duck typing on knowledge graphs. *Duck typing* is a well-known concept in dynamically typed programming languages and is based on the overall idea of loosely defining the type of an object based on its properties. Extending this concept to

KGs, without any need for type labels, we can describe the type of an entity $e \in \mathcal{E}$ in terms of its set of relations[I]. With slight abuse of notation, we will denote this set as $\mathcal{R}(e) = \{r \in \mathcal{R} \mid \exists e' \in \mathcal{E} : r(e, e') = 1\}$. An example of how the set of relations of an entity can be used to determine its type is shown in Figure 6.1. For a better qualitative analysis showing how duck typing works in real-world knowledge graphs, we refer the reader to Appendix D.1, where we show several examples taken from Wikidata (Vrandečić et al. 2014).

Relations as polar box embeddings. Inspired by region-based representations (Vendrov et al. 2016; Lai et al. 2017), and particularly by box embeddings (Vilnis et al. 2018; X. Li et al. 2019; Dasgupta et al. 2020), we represent relations as regions of the space. Our similarity function s, the dot product, is the product of the two norms of the entity and mention embeddings and the cosine of the angle between them. Since we are using this similarity to rank entities for a given mention, the norm of the mention embedding is irrelevant, whereas the entity norms encode a "prior" over entities. Therefore, we choose to represent relations as boxes in spherical polar coordinates, as shown in Figure 6.1. This representation allows guaranteeing that the cosine of the angle between two embeddings falling in the same region is constrained by the boundaries of the box. At the same time, it keeps boxes open on the radial coordinate, so as to leave the training free to use entity norms to encode prior probabilities without interference from type information. Concretely, we parameterize the box corresponding to a relation as a pair of vectors:

$$\mathit{Box}(r) = (\boldsymbol{\varphi}_r^-, \boldsymbol{\varphi}_r^+),$$

where $\boldsymbol{\varphi}_r^-, \boldsymbol{\varphi}_r^+ \in \mathbb{R}^{d-1}$ are vector of angles denoting respectively the bottom-left and top-right corners of the box in spherical coordinates. For an entity $e \in \mathcal{E}$, we say that $e \in \mathit{Box}(r)$, if the expression in polar coordinates $\boldsymbol{\varphi}_e$ of the entity representation $\boldsymbol{e}$ is between $\boldsymbol{\varphi}_r^-$ and $\boldsymbol{\varphi}_r^+$ across all dimensions. Then, our goal is to structure the latent space in such a way that $e \in \mathit{Box}(r^+)$ for every $r^+ \in \mathcal{R}(e)$ and $e \notin \mathit{Box}(r^-)$ for every $r^- \in \mathcal{R} \setminus \mathcal{R}(e)$.

6.3.2 Duck typing as an optimization problem

In order to achieve the goal mentioned above, we need to turn the intuition of Section 6.3.1 into an optimization problem. To this end, it helps to define a distance function between an entity and a box.

Entity-box distance. Following Abboud et al. (2020), who defined a similar function for box embeddings in cartesian coordinates, we define the distance between an entity

[I]More precisely, the set of relations where e is the *subject*, namely the first argument of r.

and a box as:

$$dist(e,r) = \begin{cases} \|(\boldsymbol{\varphi}_e - \bar{\boldsymbol{\varphi}}_r)/(\boldsymbol{\delta}_r + 1)\|_2 & \text{if } e \in Box(r) \\ \|(|\boldsymbol{\varphi}_e - \bar{\boldsymbol{\varphi}}_r| \circ (\boldsymbol{\delta}_r + 1) - \boldsymbol{\kappa})\|_2 & \text{otherwise,} \end{cases}$$

where $\bar{\boldsymbol{\varphi}}_r = (\boldsymbol{\varphi}_r^- + \boldsymbol{\varphi}_r^+)/2$ is the center of the box corresponding to relation r, $\boldsymbol{\delta}_r = \boldsymbol{\varphi}_r^+ - \boldsymbol{\varphi}_r^-$ is a vector containing the width of the box along each dimension, $\circ$ is the Hadamard product, / is element-wise division, and $\boldsymbol{\kappa}$ is a vector of width-dependent scaling coefficients defined as:

$$\boldsymbol{\kappa} = \frac{\boldsymbol{\delta}_r}{2} \circ (\boldsymbol{\delta}_r - \frac{1}{\boldsymbol{\delta}_r + 1} + 1).$$

Intuitively, this function heavily penalizes entities outside the box, with higher distance values and gradients, whereas it mildly pushes entities lying already inside the box towards the center. We refer the reader to Appendix D.2 for more considerations and analyses on the distance function.

Loss function for typing. To encourage an entity $e \in \mathcal{E}$ to lie inside all boxes representing the relations $\mathcal{R}(e)$ and outside the other boxes, we use a negative-sampling loss similar to the one of H. Sun, Bedrax-Weiss, et al. (2019). Our loss function is defined as:

$$\mathcal{L}_{Duck}(e) = -\mathbb{E}_{\mathrm{r}^+}[\log \sigma(\gamma - dist(e, \mathrm{r}^+))] - \mathbb{E}_{\mathrm{r}^-}[\log \sigma(dist(e, \mathrm{r}^-) - \gamma)].$$

Above, $\gamma \in \mathbb{R}$ is a margin parameter, σ is the sigmoid function, r^+ is a relation of entity e, drawn uniformly from the set of relations $\mathcal{R}(e)$, whereas r^- is a relation drawn from the set of relations $\mathcal{R} \setminus \mathcal{R}(e)$ according to the probability distribution:

$$\hat{p}(r_i^- \mid e) = \frac{\exp(-\alpha \cdot dist(e, r_i^-))}{\sum_{r_j^- \in \mathcal{R} \setminus \mathcal{R}(e)} \exp(-\alpha \cdot dist(e, r_j^-))}$$

where $\alpha \in [0, 1]$ is a temperature parameter. The lower α, the closer the distribution is to a uniform distribution, whereas higher values of α result in more weight given to boxes that are close to the entity. Notice that this objective forces the distance between an entity e and relations $r^+ \in \mathcal{R}(e)$ to be small, while keeping the entity far from boxes corresponding to the negative relations r^-. Hence, optimizing the objective $\mathcal{L}_{Duck}$ will result in clustering together entities that share many relations.

Overall optimization objective. We train the model to optimize jointly the entity-disambiguation loss of Section 6.2 and the duck-typing loss $\mathcal{L}_{Duck}$. Although we defined the loss $\mathcal{L}_{Duck}$ for entities, we calculate it for mentions as well, defining the set of relations of a mention based on the ground-truth entity $\mathcal{R}(m) = \mathcal{R}(e_m^\star)$. In order to prevent boxes from growing too large during training, we further introduce an L2 regularization term l_2 on the size of the boxes:

$$l_2 = \frac{1}{d-1}\mathbb{E}_{\mathrm{r}}[\boldsymbol{\delta}_{\mathrm{r}}^{\top}\boldsymbol{\delta}_{\mathrm{r}}].$$

Then, our final optimization objective is:

$$\mathcal{L}(m) = \mathcal{L}_{\mathit{ED}}(m) + \lambda_{\mathit{Duck}}(\mathcal{L}_{\mathit{Duck}}(e_m^{\star}) + \mathcal{L}_{\mathit{Duck}}(m) + \lambda_{l_2} l_2),$$

where $\lambda_{\mathit{Duck}}, \lambda_{l2} \in [0, 1]$ are hyperparameters defining the weight of each component of the loss.

6.4 A bi-encoder model with duck typing

Building on prior work, we used the method described in Section 6.3 to train a bi-encoder model with the same architecture of Wu et al. (2020). Compared to Wu et al. (2020), DUCK adds just a relation encoder which is only used at training time to represent relations as boxes.

6.4.1 Bi-encoder

Bi-encoders, introduced in this context by Wu et al. (2020), are a popular and efficient architecture for entity-disambiguation models based on dense retrieval. These methods rely on two different encoders f_{entity} and f_{mention} to represent entities and mentions respectively.

Entity encoder. Given a textual description of an entity $e \in \mathcal{E}$, expressed as a sequence of tokens $s_e = (s_e^{(1)}, \dots, s_e^{(|s_e|)})$, we learn an entity representation $\boldsymbol{e} \in \mathbb{R}^d$ as:

$$\boldsymbol{e} = f_{\mathit{entity}}(s_e).$$

Concretely, following prior work (Wu et al. 2020), we extract entity descriptions s_e from Wikipedia, and we structure each description s_e using the title of the Wikipedia page associated with entity e followed by the initial sentences of the body of the page, separated by a reserved token. We truncate entity descriptions s_e to a maximum sequence length of n_e. For the entity encoder f_{entity}, we used a pre-trained RoBERTa model (Liu et al. 2019), resorting to the encoding of the `[CLS]` token for the final entity representation $\boldsymbol{e}$.

Mention encoder. We model a mention as a sequence of tokens $s_m = (s_m^{(1)}, \dots, s_m^{(|s_m|)})$ denoting both the mention itself and the context surrounding it, up to a maximum mention length n_m. Following Wu et al. (2020), we used reserved tokens to denote the start and the end of a mention and separate it from the left and right context. We then calculate mention representations as:

$$\boldsymbol{m} = f_{mention}(s_m),$$

where $f_{mention}$ is a mention encoder based on a pre-trained RoBERTa model and the final mention representation $\boldsymbol{m}$ is obtained using the encoding of the `[CLS]` token. Overall, our bi-encoder is the same as the one used by Wu et al. (2020), with the only difference that we rely on RoBERTa instead of BERT (Devlin et al. 2019) as the underlying language model.

6.4.2 Relation encoder

Relation modeling. Similarly to what we described for entities in Section 6.4.1, we model a relation $r \in \mathcal{R}$ as a sequence of tokens $R = (R^{(1)}, \dots, R^{(|R|)})$. These sequences are extracted from Wikidata (Vrandečić et al. 2014), using the English label of the property and its description, separated by a reserved token. Based on R, we then compute a relation embedding $\boldsymbol{r}$ for each relation $r \in \mathcal{R}$ as:

$$\boldsymbol{r} = f_{relation}(R),$$

where $f_{relation}$ is a relation encoder similar to f_{entity} and $f_{mention}$, which computes the relation representation $\boldsymbol{r}$ as the embedding of the `[CLS]` token produced by a pre-trained RoBERTa model.

Learning boxes in polar coordinates. Given a relation representation $\boldsymbol{r}$ calculated as described above, we parametrize a box as a pair of vectors $Box(r) = (\boldsymbol{\varphi}_r^-, \boldsymbol{\varphi}_r^+)$, where:

$$\boldsymbol{\varphi}_r^- = \sigma(\mathrm{FFN}^-(\boldsymbol{r})) \cdot \pi,$$

$$\boldsymbol{\varphi}_r^+ = \boldsymbol{\varphi}_r^- + \delta_{\min} + \sigma(\mathrm{FFN}^+(\boldsymbol{r})) \cdot (\pi - \boldsymbol{\varphi}_r^- - \delta_{\min}).$$

Above, FFN^- and FFN^+ are 2-layer feed-forward networks, σ is the sigmoid function, and $\delta_{\min}$ is a margin parameter denoting the minimum width of a box across any dimension. Calculating the corners of a box in this manner allows us to achieve two main objectives: *(i)* all components of $\boldsymbol{\varphi}_r^-$ and $\boldsymbol{\varphi}_r^+$ range from 0 to π, hence they assume valid values in the spherical coordinate system, and *(ii)* $\boldsymbol{\varphi}_r^+$ is greater than $\boldsymbol{\varphi}_r^-$ across all dimensions, so that boxes are never empty and the model does not have to learn how to produce non-degenerate regions. Notice that, in a spherical coordinate system, only one of the coordinates is allowed to range from 0 to 2π, while all remaining coordinates will range from 0 to π. For simplicity, we constrain all coordinates in the interval $[0, \pi]$, thereby reducing all representations to half of the hypersphere.

6.4.3 Training and inference

Training. We train DUCK by optimizing the overall objective defined in Section 6.3. In order to compute the loss $\mathcal{L}_{Duck}$, we calculate the representations $\boldsymbol{\varphi}_e, \boldsymbol{\varphi}_m \in \mathbb{R}^{d-1}$ by converting to spherical coordinates the entity and mention representations $\boldsymbol{e}$ and $\boldsymbol{m}$ produced by the entity and mention encoders respectively. To make training more efficient, the relation representations $\boldsymbol{r}$ are pre-computed and kept fixed at training time. We use the dot product between entity and mention representations to evaluate the entity disambiguation loss $\mathcal{L}_{ED}$:

$$s(e, m) = \boldsymbol{e}^\top \boldsymbol{m}.$$

The expectations in the loss $\mathcal{L}_{Duck}$ are estimated across all relations $r^+ \in \mathcal{R}(e)$ and by sampling k relations $r^- \in \mathcal{R} \setminus \mathcal{R}(e)$ according to $\hat{p}(r^- \mid e)$. The L2 regularization on the width of the boxes is performed across all relations in a batch.

Inference. At inference time, our approach is not different from the method of Wu et al. (2020), as we only need to calculate entity and mention representations using the bi-encoder described in Section 6.4.1. We then match a mention m to the entity that maximizes the similarity function s:

$$\hat{e}_m = \arg\max_{e \in \mathcal{E}_m} s(e, m),$$

where $\mathcal{E}_m \subseteq \mathcal{E}$ is a set of candidate entities for mention m. In practice, we can precompute all entity embeddings, so that inference only requires one forward pass through the mention encoder and selecting the entity with the highest similarity.

6.5 Experiments

This section provides a thorough evaluation of our approach. First, we show that DUCK achieves new state-of-the-art results on popular datasets for entity disambiguation, closing the gap between retrieval-based methods and more expensive generative models. Then, we discuss several ablation studies, showing that incorporating type information using box embeddings in polar coordinates improves the performance of the model. Finally, we dig into qualitative analyses, showing that our model is able to place entities in the correct boxes despite the incompleteness of the information in the KG.

6.5.1 Experimental setup

In order to perform a fair comparison of DUCK with other methods, we reproduce the same experimental setup of prior work (De Cao et al. 2021; Le et al. 2019): using the

Method	AIDA	MSNBC	AQUAINT	ACE2004	CWEB	WIKI	Avg.
Dense retrieval							
Ganea et al. (2017)	92.2	93.7	88.5	88.5	77.9	77.5	86.4
Y. Yang et al. (2018)	**95.9**	92.6	89.9	88.5	**81.8**	79.2	88.0
Shahbazi et al. (2019)	93.5	92.3	90.1	88.7	78.4	79.8	87.1
X. Yang et al. (2019)	93.7	93.8	88.2	90.1	75.6	78.8	86.7
Le et al. (2019)	89.6	92.2	90.7	88.1	78.2	81.7	86.8
Fang et al. (2019)	94.3	92.8	87.5	91.2	78.5	82.8	87.9
Wu et al. (2020)†	79.6	80.0	80.3	82.5	64.2	75.5	77.0
Generative models							
De Cao et al. (2021)	93.3	94.3	89.9	90.1	77.3	87.4	88.8
Aghajanyan et al. (2022) (*CM3-Medium*)	93.5	94.2	90.1	90.4	76.5	86.9	88.6
Aghajanyan et al. (2022) (*CM3-Large*)	94.8	94.8	91.1	91.4	78.4	**88.7**	**89.8**
Type-aware models							
S. Chen et al. (2020)	93.7	94.5	89.1	90.8	78.2	81.0	86.7
Orr et al. (2021)‡	80.9	80.5	74.2	83.6	70.2	76.2	77.6
Ayoola et al. (2022) (*Wikipedia*)	87.5	94.4	**91.8**	91.6	77.8	**88.7**	88.6
Ayoola et al. (2022) (*fine-tuned*)	93.9	94.1	90.8	90.8	79.4	87.4	89.4
DUCK (*Wikipedia*)	91.0	**95.1**	91.3	**95.4**	76.9	86.1	89.3
DUCK (*fine-tuned*)	93.7	94.6	91.3	95.0	78.2	85.9	**89.8**

Table 6.1: Micro-F_1 (*InKB*) results on six entity-disambiguation datasets. **Bold** indicates the best model, underline indicates the second best results. Our results are highlighted in gray. †Model without candidate set, results from De Cao et al. (2021). ‡Results from Ayoola et al. (2022).

same datasets, the same candidate sets, and comparing the models based on the *InKB* micro-F_1 score. Following De Cao et al. (2021) and Wu et al. (2020), we train the model on the BLINK data (Wu et al. 2020), consisting of 9M mention-entity pairs extracted from Wikipedia. Entity descriptions are taken from the Wikipedia snapshot of Petroni et al. (2021). Then, we measure *in-domain* and *out-of-domain* generalization by fine-tuning the model on the training set of the AIDA-CoNLL dataset and evaluating on six test sets: **AIDA** (Hoffart et al. 2011), **MSNBC** (Cucerzan 2007), **AQUAINT** (Milne et al. 2008), **ACE2004** (Ratinov et al. 2011), **CWEB** (Gabrilovich et al. 2013) and **WIKI** (D. Guo et al. 2018).

6.5.2 Entity disambiguation results

We compared DUCK against three main categories of approaches: *(a)* methods based on *dense retrieval*, *(b) generative models*, and *(c) type-aware retrieval-based models*, namely other approaches to add type information to retrieval-based methods (DUCK pertains to this category). We report the results both for the model trained only on the BLINK data and for the model fine-tuned on AIDA, referring to the former as **DUCK** (*Wikipedia*) and to the latter as **DUCK** (*fine-tuned*).

Main results. Table 6.1 shows the results obtained by DUCK in comparison with other methods belonging to the three categories listed above. First, we notice that DUCK obtains state-of-the-art results on **MSNBC** and **ACE2004**, second best performance on

AQUAINT, and state-of-the-art results on average across all datasets. We also observe that DUCK outperforms all the other type-aware models, namely the methods of S. Chen et al. (2020), Orr et al. (2021) and Ayoola et al. (2022), showing the effectiveness of our approach to define type information and infuse it in the model. In addition, it is worth noticing that DUCK exceeds the results of recent generative models like *GENRE* (De Cao et al. 2021) and *CM3-Medium*. This is particularly impressive considering that generative models are notoriously more expensive than bi-encoder models and require one order of magnitude more time per mention at inference (Ayoola et al. 2022). Finally, we see that DUCK meets the average performance of *CM3-Large* (Aghajanyan et al. 2022), a generative model that, with its 13 billion parameters, is almost 5 times larger than *CM3-Medium* (2.7 billion parameters) and more than 18 times larger than DUCK (717 million parameters).

Knowledge-aware methods. DUCK uses a knowledge graph (Wikidata) to infuse additional information in the model. While some methods listed in Table 6.1 use indeed type information extracted from Wikidata (Ayoola et al. 2022; Orr et al. 2021; S. Chen et al. 2020), other existing knowledge-aware methods for entity disambiguation have reported results in different experimental settings, evaluating on **AIDA**, with the candidate set of Pershina et al. (2015). In order to compare with these methods, we evaluated DUCK on the candidate set of Pershina et al. (2015), and we report the results in Table 6.2. Interestingly, our model outperforms both *DeepType* (Raiman et al. 2018) and the methods of (Mulang et al. 2020) and Onoe et al. (2020). State-of-the-art results in this setting are obtained by Ayoola et al. (2022), confirming the overall good performance obtained by this method on **AIDA** in Table 6.1. However, notice that *(a)* Ayoola et al. (2022) trained the model on a custom Wikipedia dump, consisting of 100M mention-entity pairs (more than one order of magnitude larger than our dataset) and *(b)* DUCK obtains excellent results even in an out-of-domain scenario (without fine-tuning on AIDA), reaching 94.3 micro-F_1 points (an improvement of 5.1 points with respect to Ayoola et al. (2022)).

Method	**AIDA**
Onoe et al. (2020)	85.9
Raiman et al. (2018)	94.9
Mulang et al. (2020)	94.9
Ayoola et al. (2022) (*Wikipedia*)	89.1
Ayoola et al. (2022) (*fine-tuned*)	**97.1**
DUCK (*Wikipedia*)	94.3
DUCK (*fine-tuned*)	<u>96.4</u>

Table 6.2: Micro-F_1 (*InKB*) results of knowledge-aware methods on the candidate set of Pershina et al. (2015).

Method	AIDA	MSNBC	AQUAINT	ACE2004	CWEB	WIKI	Avg.
DUCK w/o types (*Wikipedia*)	85.0	93.1	87.5	87.5	73.6	84.5	85.2
DUCK w/o types (*fine-tuned*)	89.1	92.5	87.4	87.1	74.9	83.8	85.8
DUCK cartesian coord. (*Wikipedia*)	90.6	94.9	91.3	95.0	76.5	85.1	88.9
DUCK cartesian coord. (*fine-tuned*)	92.1	94.0	90.6	95.4	77.5	85.5	89.2
DUCK w/o candidate set (*Wikipedia*)	87.4	89.9	85.2	88.8	69.1	82.0	83.7
DUCK w/o candidate set (*fine-tuned*)	90.9	90.5	86.3	89.2	71.1	81.9	85.0

Table 6.3: Micro-F_1 results achieved by several ablations of DUCK

6.5.3 Ablation studies

In order to provide more insights into the performance of the model, we performed several ablation studies. **Ablations.** First, to understand the impact of duck typing on downstream entity-disambiguation performance, we performed an ablation where we removed the contribution of the $\mathcal{L}_{Duck}$ terms and the L2 regularization l_2 from the loss function (**DUCK w/o types**). In this case, we only train the model using the entity-disambiguation loss $\mathcal{L}_{ED}$, without infusing any type information. Hence, this model is equivalent to Wu et al. (2020). In addition, we assessed the benefit of using box embeddings in spherical polar coordinates by experimenting with a version of the model where boxes are expressed in cartesian coordinates (**DUCK cartesian coord**). In this case, we parametrize a box as a pair of vectors $Box(r) = (\boldsymbol{r}^-, \boldsymbol{r}^+)$, where

$$\begin{aligned} \boldsymbol{r}^- &= \mathrm{FFN}^-(\boldsymbol{r}), \\ \boldsymbol{r}^+ &= \boldsymbol{r}^- + \mathrm{ReLU}(\mathrm{FFN}^+(\boldsymbol{r})) + \delta'_{\min}. \end{aligned}$$

As before, $\delta'_{\min}$ is a margin parameter that defines the minimum width of a box, FFN^- and FFN^+ are feed-forward networks, and $\mathrm{ReLU}(x) = \max(0, x)$ is the ReLU activation function. Finally, we report the results obtained by DUCK when no candidate set is provided (**DUCK w/o candidate set**). In this case, we score each mention against the whole set of entities (which amounts to almost 6M entities).

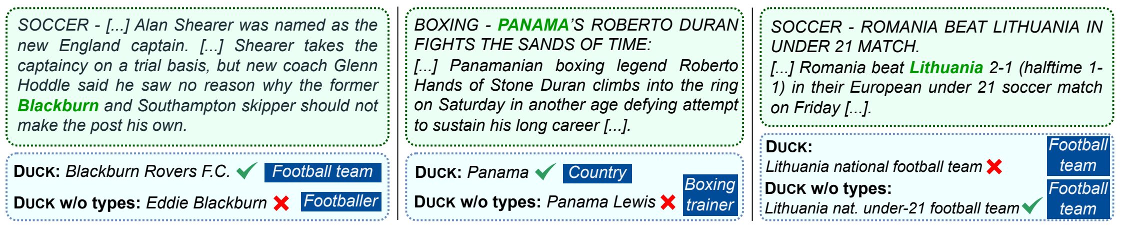
Results. Table 6.3 shows the results achieved by the ablations described above. both for the model trained on the entity-mention pairs extracted from *Wikipedia* and for the model *fine-tuned* on AIDA. Including entity types boosts the performance by approximately 4 micro-F_1 points and up to 7.9 points on **ACE2004**. The results further show the benefit of using spherical coordinates and that the model achieves good performance even without a candidate set.

6.5.4 Qualitative analyses

This section complements the quantitative results discussed so far with some qualitative analyses. First, we look into the placement of entities and boxes in the latent space,

Flag	Sport	Director	Italy	Italy national football team	Fantastic Beasts and Where to Find Them
Czech Republic	The Invincibles (en. football)	A Secret Life (film)	country	sport	orig. lang. of film or TV show
France	New England Tea Men	The Dream (1989 film)	GeoNames ID	head coach	genre
Austria	EMKA Racing	The Prodigal (1983 film)	flag image	country	publication date
Poland	Los Angeles Heroes	A Time to Sing (film)	legislative body	inception	form of creative work
Sweden	Hamline Pipers football	Modern Romance (film)	cat. for ppl. born here	cat. for memb. of team	language of work or name
Mexico	La Máquina	A Sinful Life	locator map image	Facebook ID	main subject
Germany	A. J. Foyt Enterprises	The Devil's Arithmetic (film)	Commons gallery	owned by	distributed by
D. R. of Congo	Atlético Minero	Unchained (film)	shares border with	country for sport	director
Italy	Artiach (cycling team)	As Is (film)	continent	league	cast member
Argentina	PS Barito Putera U-20	Identity Crisis (film)	named after	social media followers	spoken text audio

Table 6.4: Qualitative analysis of the boxes predicted by DUCK. The left side of the table shows the closest entities to a box, ranked according to the distance function defined in Section 6.3. The right part of the table shows the closest boxes to a given entity. Correct predictions are highlighted in green, whereas predictions that do not match relations in Wikidata are highlighted in red. Best viewed in color.



SOCCER - [...] Alan Shearer was named as the new England captain. [...] Shearer takes the captaincy on a trial basis, but new coach Glenn Hoddle said he saw no reason why the former ***Blackburn*** *and Southampton skipper should not make the post his own.*

DUCK: *Blackburn Rovers F.C.* ✓ Football team
DUCK w/o types: *Eddie Blackburn* ✗ Footballer

BOXING - ***PANAMA****'S ROBERTO DURAN FIGHTS THE SANDS OF TIME: [...] Panamanian boxing legend Roberto Hands of Stone Duran climbs into the ring on Saturday in another age defying attempt to sustain his long career [...].*

DUCK: *Panama* ✓ Country
DUCK w/o types: *Panama Lewis* ✗ Boxing trainer

SOCCER - ROMANIA BEAT LITHUANIA IN UNDER 21 MATCH. [...] Romania beat ***Lithuania*** *2-1 (halftime 1-1) in their European under 21 soccer match on Friday [...].*

DUCK: *Lithuania national football team* ✗ Football team
DUCK w/o types: *Lithuania nat. under-21 football team* ✓ Football team

Figure 6.2: Examples of the predictions of **DUCK** and **DUCK w/o types**, showing cases where **DUCK** predicts the correct entity (left and center) and where it predicts a wrong one (right). Mentions are highlighted in **bold green**.

then we discuss examples from the validation set of AIDA where type information helps DUCK in the disambiguation.

Analysis of the boxes. Table 6.4 shows a qualitative analysis of the relative placement of entities and boxes in the latent space. In the left side of the table, we looked into three boxes corresponding to the relations *flag*, *sport*, and *director*, and we reported the top 10 entities that are closer to the center of the box. The examples show a clear clustering of types, as all entities closer to the box *flag* are countries, entities inside the box *sport* are sport teams, and entities inside the box *director* are movies. For the latter box, we observe that the model predicts two movies that, in Wikidata, are missing the relation *director*, showing the ability of the model to robustly deal with incomplete information. In the right side of Table 6.4, we show which boxes are closer to three entities, namely a country, a football team and a movie, according to the distance function defined in Section 6.3. The examples show that the model is able to correctly place entities of different types in different boxes.

Examples. Figure 6.2 shows examples of the entities predicted by our method, for inputs where the prediction of DUCK differs from the ablation that does not use type information. The first two examples (left and center), clearly show how type information can help the disambiguation in cases where some keywords in the context of the mention misleads the model to making a wrong prediction. The third example (right) shows a case where DUCK predicts correctly the type of the mention, but fails to leverage some contextual information and links it to a wrong entity. This is likely due to the two entities sharing most boxes and being very close in the embedding space.

6.6 Related work

Our work builds on top of the bi-encoder architecture of Wu et al. (2020) and was partially motivated by the work of Raiman et al. (2018), who showed the benefit of using type information for entity disambiguation. Previous research has employed a variety of methods to model mentions and entities using neural networks (Z. He et al. 2013; Y. Sun et al. 2015; Yamada et al. 2016; Ganea et al. 2017; Kolitsas et al. 2018). Our method falls within a recent line of work that has proposed approaches to use type information in the disambiguation process (Raiman et al. 2018; Khalife et al. 2019; Onoe et al. 2020; S. Chen et al. 2020; Orr et al. 2021; Ayoola et al. 2022). The closest method to DUCK is the one of Ayoola et al. (2022), who used type information to improve the performance of a bi-encoder model similar to the one of Wu et al. (2020). The main difference between DUCK and the model of Ayoola et al. (2022) is that they used type labels extracted from Wikidata instead of our duck-typing approach, they represented types as points in the latent space, and they improved the performance of the model by using global entity priors (i.e., prior probabilities of an entity given a mention) extracted from count statistics. Broadly speaking, our method falls within the scope of recent research in machine learning and natural language processing (NLP) to infuse prior knowledge in neural models (Lake, Ullman, et al. 2017; Lake and Baroni 2018). Research in NLP has proposed several methods to achieve this goal, like infusing commonsense knowledge extracted from knowledge graphs (KG) in attention-based models (Bosselut et al. 2019; Keerthiram Murugesan, Atzeni, Kapanipathi, Shukla, et al. 2021), constraining attention weights in transformers using graph-structured data (Sartran et al. 2022), and improving reasoning abilities of language models with graph neural networks (Yasunaga et al. 2021; Atzeni, Bogojeska, et al. 2021).

6.7 Conclusion

This chapter introduced DUCK, a method to improve the performance of entity disambiguation models using fine-grained type information. The overall idea underlying our method was inspired by the concept of duck typing, as we loosely defined types without any need for type labels. We introduced box embeddings in spherical polar coordinates and we showed that using this form of representation allows effectively clustering entities of the same type. Crucially, we show that infusing structural information in the latent space is sufficient to close the gap between efficient methods based on dense retrieval and generative models.

Geometric reasoning in abstract symbolic tasks

Part III

7 Infusing Lattice Symmetry Priors in Attention Mechanisms

7.1 Introduction

Throughout this thesis, we have seen that infusing inductive biases and knowledge priors in neural networks is regarded as a critical step to improve their sample efficiency (Battaglia et al. 2018; Bengio 2017; Lake, Ullman, et al. 2017; Lake and Baroni 2018; Bahdanau, Murty, et al. 2019). As discussed in Section 1.3.3, there has been extensive exploration of the *Core Knowledge* priors associated with human intelligence in developmental science (Spelke et al. 2007). This theory posits that humans possess a limited set of distinct systems of core knowledge, forming foundational elements upon which new adaptable skills and belief systems can be constructed. In the realm of artificial intelligence, recent studies propose integrating these same priors into AI systems (Chollet 2019). However, it remains unclear how to incorporate these priors into neural networks.

The Abstraction and Reasoning Corpus (ARC) (Chollet 2019) was introduced as an AI benchmark that requires models that can leverage the *Core Knowledge* priors of developmental science to learn more efficiently. According to Chollet (2019), constructing algorithms grounded in these priors represents an essential step for general AI progress. As claimed by Chollet (2019) and confirmed by our experiments, the ARC dataset requires models that are extremely efficient and cannot be meaningfully undertaken using popular architectures like transformers and attention.

An important category of *Core Knowledge* priors includes *geometry and topology priors.* Indeed, significant attention has been devoted to incorporating such priors in deep learning architectures by rendering neural networks invariant (or equivariant) to transformations represented through group actions (Bronstein et al. 2021). However, group-invariant learning helps to build models that systematically *ignore* specific transformations applied to the input (such as translations or rotations).

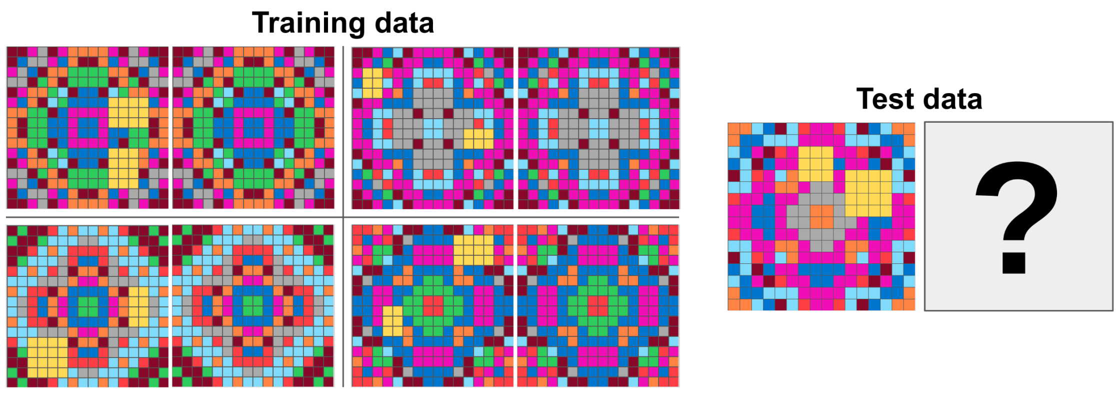


Figure 7.1: We consider problems that involve learning a geometric transformation on the input data as a sub-problem. The displayed task (taken from ARC) entails learning to map, for each pair, the left to the right image. We investigate how to solve such tasks more sample-efficiently by imbuing self-attention with the ability to exploit lattice symmetry priors.

We take a complementary perspective and aim to help neural networks to learn functions that incorporate geometric transformations of their input (rather than to be invariant to such transformations). In particular, we focus on group actions that belong to the symmetry group of a lattice. These transformations are pervasive in machine learning applications, as basic transformations of sequences, images, and other higher-dimensional regular grids fall in this category. While attention and transformers can in principle learn these kind of group actions, we show that they require a significant amount of training data to do so.

To address this sample complexity issue, we introduce LatFormer, a model that relies on attention masks in order to learn actions belonging to the symmetry group of a lattice, such as translation, rotation, reflection, and scaling, in a *differentiable* manner. We show that, for any such action, there exists an attention mask such that an untrained self-attention mechanism initialized to the identity function performs that action. We further prove that these attention masks can be expressed as convolutions of the identity, which motivates a modification to the standard attention module where the attention weights are modulated by a mask generated by a convolutional neural network (CNN).

Our experiments focus on abstract geometric reasoning and, more specifically, on ARC and its variants, as they are widely regarded as challenging benchmarks for machine learning models (Acquaviva et al. 2021; Chollet 2019). On these datasets, we aim to reduce the gap between neural networks and hand-engineered search algorithms. To probe the sample efficiency of our method, we compared its ability to learn synthetic geometric transformations against Transformers and attention modules. Then, we annotated ARC tasks based on the knowledge priors they require, and we evaluated LatFormer on the ARC tasks requiring geometric knowledge priors. Finally, we performed experiments on the more recent *Language-complete ARC* (LARC) (Acquaviva et al. 2021), which enriches ARC tasks with natural-language descriptions, and we compared our model against strong baselines based on neural program synthesis. Our results provide evidence that LATFORMER can learn geometric transformations with 2 orders of magnitude fewer training data than transformers and attention. We also significantly reduce the gap between neural and classical approaches on ARC, providing the first neural network that reaches good performance on ARC tasks requiring geometric knowledge priors.

7.2 Formalizing the group-action learning problem

To build intuition on the kind of basic priors that we aim to infuse in neural networks, Figure 7.1 shows a task borrowed from ARC (Chollet 2019). The task entails learning to fill out the yellow patches in the leftmost image (input) so that the resulting image satisfies a 90° rotation symmetry. The learner is given only a small set of input-output pairs: the ARC tasks have 3.3 training examples on average. Though the task is challenging for a general neural network (due to the small number of examples), it becomes easier under the prior knowledge of discrete two-dimensional point groups, one of which is the cyclic group of 4-fold rotations C_4. Under this prior, it can be solved by the composition of a group action (rotating each image x by some $\mathsf{g} \in \mathsf{C}_4$) and a shallow neural network with a non-linear activation (mapping yellow to zero and taking a pixel-wise max).

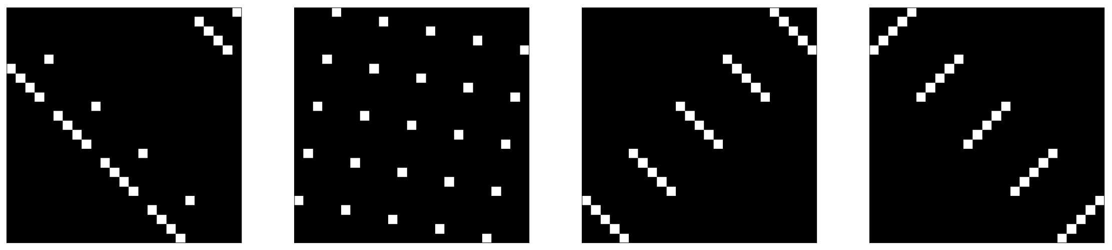
More formally, we are interested in helping neural networks learn lattice transformations in a sample efficient manner by infusing knowledge priors in the model. Motivated by ARC, we focus on learning geometric transformations that belong to the symmetry group of a lattice. This pertains to the more general problem of learning group actions given the input and the output of the transformation. We refer to this problem as group-action learning.

Concretely, we consider input-output transformations involving a group element g taken from some known group G that can be expressed under the general formulation:

$$y = f(\mathsf{g} \circ x, x), \quad \mathsf{g} = g(x) \in \mathsf{G} \qquad \text{(\textit{group-action learning})}.$$

Above, $x \in \mathbb{R}^{d_{\text{in}}}$ and $y \in \mathbb{R}^{d_{\text{out}}}$ are input and output examples, f, g are unknown functions, and $\circ$ denotes the application of a group action. The group element g can depend on the input data itself. More generally, even the function f could depend on more than one transformation of x, with actions belonging to various groups of interest.

It is important to stress that group-action learning is the exact antithesis of the typical



(a) Translation by (1, 1) **(b)** Rotation by 90° **(c)** Vertical reflection **(d)** Horiz. reflection

Figure 7.2: Examples of attention masks implementing transformations in two dimensions, including: (a) translation by 1 pixel on both axes, (b) rotation by $90°$ counterclockwise, (c) vertical reflection and (d) horizontal reflection around the center. White represents value 1 and black 0.

group invariant and equivariant learning problems (Bronstein et al. 2021):

$$\begin{aligned} y &= f(\mathrm{g} \circ x) \quad \text{for } \textit{every } \mathrm{g} \in \mathrm{G} \qquad (\textit{invariant learning}), \\ \mathrm{g} \circ y &= f(\mathrm{g} \circ x) \quad \text{for } \textit{every } \mathrm{g} \in \mathrm{G} \qquad (\textit{equivariant learning}). \end{aligned}$$

Intuitively speaking, whereas in group-action learning one aims to learn functions that involve specific (and data-dependent) transformations of our data by actions of the group, in in/equivariant learning the goal is to build models that are oblivious to such actions in a systematic manner.

7.3 Attention masks for core geometry priors

This section prepares some theoretical grounding for LatFormer, our approach to learn the transformations for lattice symmetry groups in the form of attention masks. The section defines attention masks and explains how they can be leveraged to incorporate geometry priors when solving group action learning problems on sequences and images.

7.3.1 Modulating attention weights with soft masking

Consider the scaled dot-product attention mechanism as defined in Vaswani et al. (2017). In our formulation, we consider real-valued masks $\boldsymbol{M} \in [0,1]^{n_Q \times n_K}$ that rescale attention weights:

$$\boldsymbol{A} = \mathrm{softmax}\Big(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}\Big) \odot \boldsymbol{M},$$

where $\boldsymbol{Q} \in \mathbb{R}^{n_Q \times d}$ is the query parameter of the attention mechanism, $\boldsymbol{K} \in \mathbb{R}^{n_K \times d}$ is the key, d is the dimensionality of the model, n_Q and n_K are the sizes of the sets encoded by the query and key matrices respectively, and $\odot$ is the Hadamard product. Attention masks have been widely used to constrain the values of the attention weights and are usually binary masks applied before the $\mathrm{softmax}$ activation (Vaswani et al. 2017; Sartran et al. 2022). However, as we aim to learn $\boldsymbol{M}$, we apply the mask after the $\mathrm{softmax}$ operation in order to avoid squashing the gradient. Therefore, we rescale the attention weights to sum to 1 when calculating the output $\boldsymbol{X}$ of the attention mechanism:

$$\mathrm{MaskedAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}; \boldsymbol{M}) = \frac{\boldsymbol{A}}{\boldsymbol{A} \cdot \mathbf{1}_{n_K} \mathbf{1}_{n_K}^{\top}} \boldsymbol{V},$$

with $\mathbf{1}_n$ being a vector of ones of size n and $\boldsymbol{V} \in \mathbb{R}^{d \times n_K}$ being the value parameter of the attention mechanism. Though masking can also be applied in cross-attention, in the following we primarily focus on *self-attention*, where $\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V} = \boldsymbol{X}$. For ease of notation, we write $\mathrm{MaskedAttention}(\boldsymbol{X}; \boldsymbol{M})$ whenever the query, key and value are the same matrix $\boldsymbol{X}$.

7.3.2 Existence of attention masks for lattice symmetry actions

This section discusses group actions that can be represented by attention masks. To develop intuition, let us first consider the simple example of translation in a one-dimensional lattice. Supposing that $\boldsymbol{x} = (x_1, \ldots, x_n)^{\top}$ is a vector of n elements, we have:

$$\mathrm{MaskedAttention}(\boldsymbol{x}; \boldsymbol{M}) = (x_n, x_1, \ldots, x_{n-1})^{\top}$$

with:

$$\boldsymbol{M} = \begin{pmatrix} 0 & 0 & \cdots & 1 \\ 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix}.$$

Hence, when $\boldsymbol{M}$ is the circulant permutation matrix shown above, we have that the mask shifts the input $\boldsymbol{x}$ by one element to the right.

Beyond translation, it is natural to ask what kinds of group actions we can perform with attention masks on data with a more high-dimensional topological structure. The following theorem provides existence statements for data whose underlying topological space is a hypercubic lattice (such as sequences, images and higher-dimensional regular grids).

Theorem 7.3.1 (Existence)**.** *Let G_m be the symmetry group of the m-dimensional hypercubic lattice, including translational symmetry, 4-fold rotational symmetry and vertical, horizontal and diagonal reflections. Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be a vectorized representation of an m-dimensional tensor $\mathsf{X} \in \mathbb{R}^{l_1 \times \cdots \times l_m}$, with $n = l_1 \cdot \ldots \cdot l_m$. For any group action $\mathsf{g} \in \mathsf{G}_m$,*

Transformation	**Fourier shift**	**Size of X**
Identity	$\boldsymbol{o}(\mathrm{g}_i)_k = 0$	$n = l_1$
Translation (by δ)	$\boldsymbol{o}(\mathrm{g}_i)_k = -\delta$	$n = l_1$
Reflection	$\boldsymbol{o}(\mathrm{g}_i)_1 = (n-1)$, $\boldsymbol{o}(\mathrm{g})_k = \boldsymbol{o}(\mathrm{g})_{k-1} - 2$	$n = l_1$
Rotation ($90°$)	$\boldsymbol{o}(\mathrm{g}_i)_k = k \cdot (l_1 - 1) - \lfloor (k-1)/l_1 \rfloor$	$n = l_1 \times l_2$
Upscaling (by h)	$\boldsymbol{o}(\mathrm{g}_i)_k = (k - 1 \bmod h) + (h-1) \cdot \lfloor (k-1)/h \rfloor$	$n = l_1$

Table 7.1: Fourier shifts for the transformations on the 1-dimensional and square lattices. We denote with $\boldsymbol{o}(\mathrm{g}_i)_k$ the k-th component of the vector $\boldsymbol{o}(\mathrm{g}_i) \in \mathbb{R}^n$, for $k = 1, \dots, n$. As stated in Theorem 7.3.2, attention masks for higher-dimensional lattices can be obtained by the Kronecker product of primitive masks defined over the 1-dimensional and square lattices. Composition of actions is given by matrix multiplication of the masks.

there exists an attention mask $\boldsymbol{M}_\mathrm{g} \in \{0,1\}^{n \times n}$, such that:

$$\mathrm{MaskedAttention}(\boldsymbol{X}; \boldsymbol{M}_\mathrm{g}) = \mathrm{g} \circ \boldsymbol{X}.$$

In other words, Theorem 7.3.1 states that any translation, rotation or reflection can be expressed in terms of an attention mask. Figure 7.2 shows some examples of masks corresponding to translation, rotation and reflection operations on square lattices.

In the following, we adopt the convention of writing $\boldsymbol{M}_\mathrm{g}$ to mean the mask that implements action g. For more details and for a proof of Theorem 7.3.1, we refer the reader to Appendix E.3.

7.3.3 Representing attention masks for lattice transformations

To facilitate the learning of lattice symmetries, one needs to determine methods to parameterize the set of feasible group elements. Fortunately, as precised in the following theorem, the attention masks considered in Theorem 7.3.1 can be expressed conveniently under the same general formulation.

Theorem 7.3.2 (Representation)**.** *Let G_m be the symmetry group of the m-dimensional hypercubic lattice and $\mathrm{g} \in \mathsf{G}_m$ be an action on a tensor $\mathsf{X} \in \mathbb{R}^{l_1 \times \cdots \times l_m}$. Then, there exist some primitive attention masks $\boldsymbol{M}_{\mathrm{g}_i} \in \{0,1\}^{n_i \times n_i}$ such that*

$$\boldsymbol{M}_\mathrm{g} = \bigotimes_i \boldsymbol{M}_{\mathrm{g}_i} \quad \textit{and}$$

$$\mathcal{F}(\boldsymbol{M}_{\mathrm{g}_i}) = \mathcal{F}(\boldsymbol{I}_{n_i}) \, \exp(-\frac{2\pi j}{n_i} \, \boldsymbol{o}(\mathrm{g}_i) \, \boldsymbol{r}_{n_i}^\top),$$

where $\boldsymbol{M}_{\mathrm{g}} \in \{0,1\}^{n\times n}$ is an attention mask implementing g, $\mathrm{g}_i \in \mathsf{G}_{m_i}$ *for some* $m_i \in \{1,2\}$ *is an action on the one-dimensional or square lattice,* $\otimes$ *is the Kronecker product,* $\mathcal{F}$ *is the Fourier transform applied column-wise,* $\boldsymbol{I}_{n_i}$ *is the* $n_i \times n_i$ *identity matrix,* j *is the imaginary unit,* $\boldsymbol{r}_{n_i} = (1,2,\ldots,n_i)^\top$*, and* $\boldsymbol{o}(\mathrm{g}_i)$ *is defined as in Table 7.1.*

To obtain an intuitive understanding of Theorem 7.3.2, it helps to revisit the example of translation by $\delta = 1$ of a sequence $\boldsymbol{x} \in \mathbb{R}^n$ on the 1-dimensional lattice ($m = 1$). Consulting Table 7.1, we find that $\boldsymbol{o}(\mathrm{g})$ is a vector containing -1 at every position and we know $\boldsymbol{M}_{\mathrm{g}}$ is the permutation circulant matrix of Section 7.3.2. Indeed, by the time-shifting property of the Fourier transform, $\boldsymbol{M}_{\mathrm{g}}$ can be obtained by shifting the rows of the identity by -1. In general, vector $\boldsymbol{o}(\mathrm{g})$ has a convenient intuitive interpretation as its k-th component represents the relative position (with respect to k) of the element that the k-th row of $\boldsymbol{X}$ attends to. For instance, in the one-dimensional example of translation by one element to the right, each element attends to the one immediately before. Hence, we have $\boldsymbol{o}(\mathrm{g})_k = -1$ for any $k = 1,\ldots,n$.

For higher-dimensional lattices, attention masks can be expressed as the Kronecker product of the attention masks for lower-dimensional cases. For instance, on the square lattice, a translation by 1 pixel on both dimensions is the Kronecker product of the two circulant matrices corresponding to a translation by 1 pixel on the one-dimensional lattice, as shown in Figure 7.2a. On more than one dimension, we can additionally define 4-fold rotations, still following the same formulation, with $\boldsymbol{o}(\mathrm{g}_i)$ defined as in Table 7.1.

Although strictly not a symmetry operation, *scaling transformations* of the lattice can also be defined in terms of attention masks under the same general formulation of Theorem 7.3.2, as reported in Table 7.1. Therefore, for completeness, we will consider scaling transformations as well in our experiments.

Notice that Theorem 7.3.2 allows us to derive a way to calculate the attention masks. In particular, we can express our attention masks as a convolution operation on the identity, as stated below.

Corollary 7.3.3. *Let* G_m *be the symmetry group of the* m*-dimensional hypercubic lattice and let* $\boldsymbol{M}_{\mathrm{g}} \in \mathbb{R}^{n\times n}$ *be an attention mask implementing action* $\mathrm{g} \in \mathsf{G}_m$*. Then:*

$$\boldsymbol{M}_{\mathrm{g}}[:,i] = \mathcal{F}^{-1}(\exp(-\frac{2\pi j}{n} \cdot \boldsymbol{o}(\mathrm{g}) \cdot \boldsymbol{r}_n^\top))[:,i] \star \boldsymbol{I}_n[:,i],$$

where $\star$ *denotes the convolution operation.*

In other words, we can represent any mask in our framework as the Kronecker product of convolutions of the identity matrix with predefined kernels. This motivates us to design a convolutional neural network that produces our attention masks by successive convolutions of the identity.

7.4 The LatFormer architecture

While in principle the problem of inferring group actions from input-output pairs can be solved via search over finite groups, in practice the size of the group for lattice symmetry actions makes this approach unfeasible[I]. Moreover, we are interested in learning unknown functions jointly with the transformation, which cannot be solved by searching on the space of group actions. Using a neural agent to search the space of possible actions would be a viable alternative, but this would make the problem non-differentiable and we would need to resort to reinforcement learning methods.

In this work, we aim to solve the problem in a *differentiable* way. Inspired by the observations above, we introduce LatFormer, which incorporates the insights of Section 7.3 into a neural architecture. We propose to use gated CNNs to parameterize the masks and we introduce an additional smoothing technique for easier optimization.

7.4.1 Lattice mask experts as convolutional networks

Attention modules in neural networks usually include an attention mechanism with learnable linear transformations of the inputs[II] followed by a feed-forward network (FFN), as in the Transformer encoder layer (Vaswani et al. 2017).

To infuse core geometry priors in the attention module, we propose to modulate the attention weights with a mask generated by an additional layer, as shown in Figure 7.3a. We refer to this layer as *Lattice mask expert,* as it specializes towards specific transformations of the lattice. To understand the purpose of this layer, it is useful to remember that, by the analysis conducted in Section 7.3, even if the attention and FFN layers are initialized to the identity function, the mask expert can generate attention masks that produce precise geometric transformations of the input.

By Corollary 7.3.3, we know that each group action on the lattice can be represented by a mask that is a convolution of the identity and we have an analytical expression to calculate the kernels of the convolution. We can leverage this notion to design CNNs that produce attention masks corresponding to specific group actions by following the general formulation:

$$\begin{aligned} \boldsymbol{M}_0 &= \boldsymbol{I}, \\ \boldsymbol{M}_{l+1} &= \alpha_l \,\mathrm{Conv}(\boldsymbol{M}_l, \boldsymbol{K}_l) + (1-\alpha_l)\boldsymbol{M}_l \ \text{ for } \ l = 0, \dots, L-1, \end{aligned}$$

Above, $\boldsymbol{M}_L$ is the predicted mask, $\alpha_l = \sigma_l(\boldsymbol{X};\boldsymbol{\theta}) = \mathit{FFN}_l(\boldsymbol{X},\boldsymbol{\theta})$ is the output of a gating function, $\boldsymbol{\theta}$ is a learnable parameter, and $\boldsymbol{K}_l$ is the kernel of the l-th convolutional layer whose weights are determined based on Corollary 7.3.3 and Table 7.1.

[I]The size of the groups we consider grows with a polynomial of n and exponentially with m.
[II]For simplicity, we omitted linear transformations in the definition of MaskedAttention.

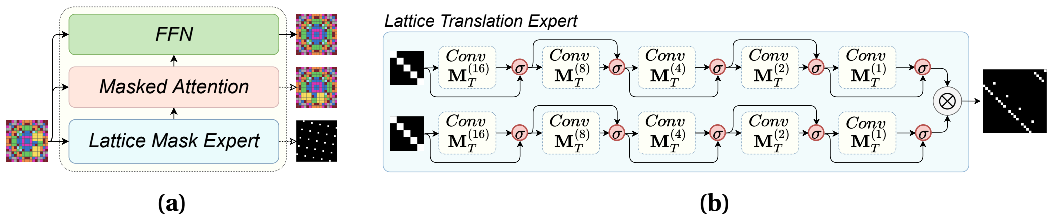

FFN
Masked Attention
Lattice Mask Expert
Lattice Translation Expert
Conv $\mathbf{M}_T^{(16)}$
Conv $\mathbf{M}_T^{(8)}$
Conv $\mathbf{M}_T^{(4)}$
Conv $\mathbf{M}_T^{(2)}$
Conv $\mathbf{M}_T^{(1)}$
σ
$\otimes$
(a)
(b)

Figure 7.3: A LATFORMER layer (a) and an architecture for a *Lattice translation expert* (b). The LATFORMER layer (a) is a standard Transformer encoder layer augmented with a *Lattice mask expert* constrained to generate attention masks corresponding to a geometric transformation of the input. The *Lattice translation expert* (b) is a particular instance of a *Lattice mask expert* that produces translation masks. In the architecture above, every convolutional layer is meant to shift the input by a power of 2 and can be skipped by a gating function (denoted as σ).

As an example, Figure 7.3b shows an architecture that generates translation masks. Following Theorem 7.3.2, the expert computes the translations along the two dimensions separately and then aggregates the resulting masks doing the Kronecker product. Hence, a *Lattice translation expert* with L convolutional layers for each dimension can generate any translation up to $\delta = 2^L - 1$ elements per dimension. At inference time, the values of the gates can be discretized, in such a way that the generated mask provably performs a meaningful group action.

Similarly to the expert in Figure 7.3b, we can define gated CNNs for rotation, reflection, and scaling. The product of experts (i.e., the combination of more actions) can be obtained by either chaining the experts or multiplying the attention masks generated by different experts. For more details, we refer the reader to Appendix E.1.

7.4.2 Mask smoothing for easier training

The framework described so far parameterizes discrete transformations of a lattice in a differentiable manner. Nevertheless, to improve the training of LATFORMER, we found it beneficial to also apply a smoothing operation on the attention masks. Our approach entails defining an adjacency relation between group elements and applying graph convolution with a heat kernel on the corresponding graph. This encourages the optimizer to favor weight updates that change the masks in a smooth manner with respect to the geodesic distance implied by the graph. Concretely, we define the neighbors of each element g_i on the lattice as those $\mathsf{g}_j = \mathsf{e} \circ \mathsf{g}_i$ reachable by an application of a primitive action e, such as translation by a single pixel in one dimension, rotation by $90°$, and vertical/horizontal reflection. The notion of neighborhood gives rise to a graph whose vertex set is the lattice group and that contains one edge for every pair of neighboring actions.

As before, it helps to consider different kinds of transformations separately. For instance, as shown in Figure 7.4, for 2D rotations the underlying graph is a cycle with 4

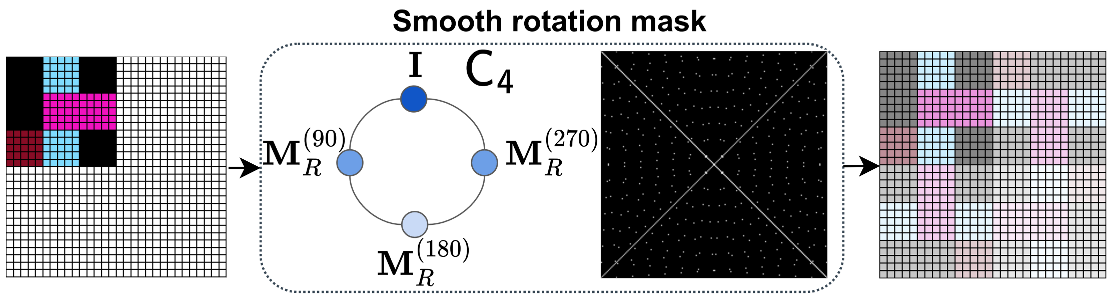


Figure 7.4: Rotational smoothing can be obtained by heat diffusion over the cyclic graph of rotation masks.

elements due to the underlying point group for 4-fold rotations being the cyclic group C_4. Performing heat diffusion can be achieved by repeated neighborhood averaging over the cycle and yields a smoothed rotation mask that performs all rotations at the same time (rightmost image in Figure 7.4). We can extend the same approach to all lattice transformations: for instance, in the case of translation, the underlying graph is a grid and the smoothing operation is akin to convolution with a Gaussian kernel.

To train LatFormer with smoothed masks, we compute two predictions: one with the non-smooth mask predicted by the model and one with a smoothed version of the same mask. The final loss is the sum of two cross-entropy losses calculated separately for the two predictions.

7.5 Experiments

To evaluate our method, we first developed a set of synthetic tasks in order to compare LatFormer to attention modules and Transformers with respect to sample efficiency in learning basic geometric transformations. Then, we annotated the ARC tasks based on the knowledge priors they require, and we assessed the performance of our method on this challenging dataset. Finally, we experimented with the LARC (Acquaviva et al. 2021) dataset and compared our method to stronger baselines based on neural program synthesis. We report additional experimental results in Appendix E.2.4.

7.5.1 Sample efficiency on geometric transformations

As a preliminary study, we probed the ability of LatFormer to learn geometric transformations efficiently. To this end, we compared the performance of our model to a transformer (Vaswani et al. 2017) and an attention module (the same architecture as our approach, without the mask expert) on synthetic tasks with increasing number of examples. Inspired by ARC, we generated a set tasks where the model needs to infer

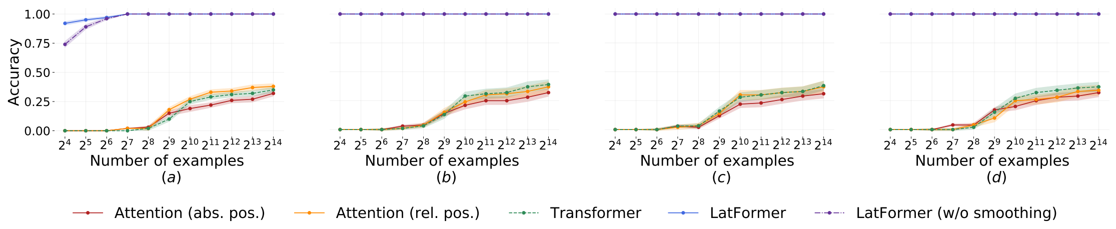


Figure 7.5: Sample efficiency of our method compared to the baselines on synthetic tasks on translation (a), rotation (b), reflection (c) and scaling (d). The y axis denotes the mean accuracy across tasks belonging to the same category, whereas the error shade is the standard deviation.

a geometric transformation from input-output pairs. The input is a grid taken from the ARC tasks and the output is either a translation, rotation, reflection or scaling of the input. The specific transformation applied to the input grid defines the task and is consistent across all examples in the same task.

We evaluated the models based on the mean accuracy across tasks. Figure 7.5 shows the accuracy of our model compared to the baselines and to a version of LatFormer without smoothing. The plots show that LatFormer can generalize better and from fewer examples than transformers and attention modules both with absolute positional encodings (Vaswani et al. 2017) and relative positional encodings (Shaw et al. 2018). Additionally, our results show that the smoothing operation described in Section 7.4.2 is helpful for larger groups. More details are reported in Appendix E.2.1.

7.5.2 Geometric reasoning on ARC tasks

To assess the ability of our approach to learn efficiently on a more challenging use case, we focused on a subset of the ARC dataset (Chollet 2019) requiring geometric priors for which our method could be a principled solution. To this end, we annotated the ARC tasks based on the knowledge priors they require, using the list of priors provided by Chollet (2019) as a reference. Appendix E.2.2 provides more details about the annotation of ARC and Figure E.2a in the Appendix shows the knowledge priors that we considered and their distribution across the ARC tasks.

We assessed the performance of our model on the tasks that require only the geometric transformations that we addressed in this work, namely translation, rotation, reflection and scaling. Table 7.2 shows our results compared to neural baselines, including CNNs, attention with relative positional encodings (Shaw et al. 2018), PixelCNN (Gul et al. 2019), and Transformers (Vaswani et al. 2017), and a Differentiable Neural Computer (Graves, Wayne, Reynolds, et al. 2016) with spectral regularization (Kolev et al. 2020). We additionally compared to a Transformer model that has access to precomputed transformations of the input (**Transformer + data augmentation**). Precomputing all group actions is only feasible for smaller groups (rotation, reflection and scaling).

	Translate	Rotate + Translate	Reflect + Translate	Scale + Translate
CNN	0.019	0.000	0.000	0.000
Attention (abs. pos.)	0.019	0.000	0.023	0.000
Attention (rel. pos.)	0.019	0.000	0.023	0.000
PixelCNN	0.019	0.000	0.000	0.000
Transformer	0.038	0.000	0.045	0.000
Differentiable Neural Computer	0.038	0.000	0.045	0.000
Transformer + data augmentation	-	0.200	0.184	0.091
LatFormer	0.365	**0.800**	0.591	0.545
Search over hand-crafted DSL	**0.673**	0.400	**0.614**	**0.727**

Table 7.2: Performance on ARC tasks that involve lattice symmetry priors.

Further, Table 7.2 reports the performance obtained by a search algorithm applied on top of a hand-engineered domain-specific language (DSL). This approach searches all possible programs in the DSL that can map the input grids to the corresponding output grids successfully. We use the implementation of Wind (2020), which obtained the best results at the ARC Kaggle competition out of almost 1000 submissions [III]. This approach does not use any learnable component and the results are provided as a reference. We notice that LatFormer significantly reduces the gap between neural networks and the current best approach for ARC, even outperforming the search algorithm for one category of tasks.

Though we restrict to only a subset of the tasks and there is definitely room for improvement even on these tasks, we reach considerably better performance than the baselines. Therefore, we believe our results advocate for the applicability of end-to-end differentiable models even on problems requiring sample-efficient abstract reasoning. To the extent of our knowledge, this is the first evidence of a neural network achieving this performance on ARC tasks.

7.5.3 Comparison with neural program synthesis

Recently, Acquaviva et al. (2021) introduced the *Language-complete Abstraction and Reasoning Corpus* (LARC), which provides natural language descriptions of 88% of the ARC tasks, generated by human participants who where asked to communicate to other humans a set of precise instructions to solve a task.

Acquaviva et al. (2021) evaluated several models based on neural program synthesis on LARC. All models generate symbolic programs from a carefully designed domain-specific language (DSL) following a *generate-and-check* strategy. First a neural model generates a program from the grammar of the DSL (Ellis et al. 2020) and then then program is checked against the input-output pairs to ensure that it can generate all training examples.

[III] https://www.kaggle.com/competitions/abstraction-and-reasoning-challenge/

	Translate	Rotate + Translate	Reflect + Translate	Scale + Translate
LARC (IO)	0.17	0.00	0.42	**1.00**
LARC (IO + NL)	0.17	0.00	0.42	**1.00**
LARC (IO + NL pseudo)	0.25	0.00	0.42	**1.00**
LatFormer	**0.33**	**1.00**	0.50	**1.00**
LatFormer + NL	**0.33**	**1.00**	**0.58**	**1.00**

Table 7.3: Comparison of LatFormer with neural program synthesis methods with access to both input-output pairs and natural language descriptions on LARC

We compare against the following baselines identified by Acquaviva et al. (2021). **LARC (IO)** is a model that has only access to input-output pairs, as our LatFormer. **LARC (IO + NL)** has access to the natural language descriptions as well and uses a pre-trained T5 model (Raffel et al. 2020) to represent the text. **LARC (IO + NL pseudo)** uses *pseudo-annotation* to encourage the learning of compositional relationships between language and programs: during training, the model is given additional synthetic language-to-program pairs generated by annotating primitive examples in the DSL with linguistic comments. We refer the reader to Appendix E.2.3 for more details.

In order to compare to the work of Acquaviva et al. (2021), we evaluated their models on the set of LARC tasks that correspond to ARC tasks in our subset requiring geometric knowledge priors. Additionally, following Acquaviva et al. (2021) we allowed LatFormer to access the textual descriptions by using a pre-trained T5 model to generate a representation of the text. This embedding is provided as input both to the *Lattice Mask Expert* and the *FFN* layers of LatFormer (**LatFormer + NL**). Table 7.3 shows the results of our experiments on the LARC dataset. The program-synthesis methods require a training stage on a portion of the tasks. Therefore, the LatFormer models where only evaluated on the same testing tasks of LARC, using the same train-test split of Acquaviva et al. (2021). Overall, our results show that LatFormer performs better than program synthesis on the subset of tasks requiring geometric priors, with no need for a carefully designed DSL. This advantage comes to the expense of being restricted to tasks involving geometric priors, whereas program-synthesis approaches can be used on a wider set of tasks. We also observe that the natural language descriptions marginally helped our model on one category of tasks. Our findings corroborate with Acquaviva et al. (2021) in this remark.

7.6 Related work

Our work was inspired by a previous investigation of self-attention layers which identified sufficient conditions such that they can perform convolution when equipped with relative positional encodings (J.-B. Cordonnier et al. 2020; Andreoli 2019). Rather than relying on relative encodings, we here show how soft-masking can be used to learn sample efficiently more general input transformations.

To the extent of our knowledge, the group-action learning problem has not been explicitly and generally formulated in previous work. That being said, many previous works have focused on specific instances, such as learning to sort (Graves, Wayne, and Danihelka 2014; Reed et al. 2015; Y. Li et al. 2020) by selecting an element of the permutation group S_n, docking/folding by roto-translating objects according to an action in the special Euclidean group $\mathsf{SE}(3)$ (Sverrisson et al. 2022; Stärk et al. 2022; Jumper et al. 2021), and graph spectrum generation where the learned actions belong to the Stiefel manifold (Martinkus et al. 2022).

Our work is similar in spirit to recent efforts in neuro-symbolic visual reasoning (Justin Johnson, Hariharan, Maaten, Hoffman, et al. 2017; Justin Johnson, Hariharan, Maaten, Fei-Fei, et al. 2017; Y. Goyal et al. 2017; Mao et al. 2019; Higgins et al. 2018). Many approaches based on attention mechanisms have been proposed in the past few years (Hudson et al. 2018; Hudson et al. 2019). Our work differentiates from previous lines of research in that we aim to learn basic geometric reasoning in a sample-efficient way, rather than modeling relationships between high-level concepts.

Finally, some recent works came to our same conclusion on the advantages of using attention masks to incorporate prior knowledge in neural networks. As an example, Yan et al. (2020) focus on the task of learning subroutines (e.g., sorting algorithms) and use a CNN to generate an attention mask for a Transformer encoder. They show that learning the attention mask allows them to generalize to longer sequences than the ones provided at training time. Similarly, Sartran et al. (2022) used precomputed attention masks to incorporate syntactic biases in language models.

7.7 Conclusion

Motivated by the long-term ambitious goal of infusing core knowledge priors in neural networks, this chapter focused on how to help deep learning models to learn geometric transformations efficiently. Specifically, we proposed to incorporate lattice symmetry biases into attention mechanisms by modulating the attention weights using learned soft masks. We have shown that attention masks implementing the actions of the symmetry group of a hypercubic lattice exist, and we provided a way to represent these masks. This motivated us to introduce LatFormer, a model that generates attention masks corresponding to lattice symmetry priors using a CNN. Our results on synthetic tasks show that our model can generalize better than the same attention modules without masking and Transformers. Moreover, the performance of our method on a subset of ARC provides the first evidence that deep learning can be used on this dataset, which is widely considered as an important open challenge for AI research.

Conclusion

8 Conclusion and Future Work

8.1 Conclusion

In this thesis, we have explored the reasoning abilities of neural models across a variety of tasks and we have proposed methods to improve their sample efficiency and generalization performance by leveraging proper knowledge priors.

In Part I, we focused on reasoning in complex interactive environments requiring prior knowledge about the world, language understanding, and look-ahead planning. In Chapter 2, we learned that providing prior knowledge structured as a graph can significantly improve sample efficiency and generalization performance. Interestingly, we observed this behavior when using a knowledge graph to infuse simple commonsense knowledge in the model, even though recent work (Davison et al. 2019; Bosselut et al. 2019; Cui et al. 2021) has shown that pre-trained language models (which in our case where used to encode observations and actions) can directly identify commonsense facts. This confirms findings form other works (B. Y. Lin et al. 2019; Klein et al. 2019; Kassner and Schütze 2020), which argued that structured commonsense reasoning is not captured well by pretrained language models like BERT (Devlin et al. 2019) and RoBERTa (Liu et al. 2019).

The observations of Chapter 2 suggest that graph-based representations can be a good inductive bias for symbolic reasoning. In order to verify this hypothesis, Chapter 3 further enhanced the model of Chapter 2 with a knowledge graph that is constructed from the natural language observations provided by the environment. Notice that the model of Chapter 2 already encodes this information, but using a graph-based representation results in improved sample efficiency and higher generalization performance. This confirms that the results of Chapter 2 do not stem only from the additional knowledge but rather from the use of graph-structured representations as an inductive bias.

To conclude Part I, in Chapter 4 we proposed a method inspired by case-based reasoning (Aamodt et al. 1994) to train the agent of Chapter 3, reaching dramatic improvements in both sample efficiency and out-of-distribution generalization. This shows that existing reasoning algorithms can be used to guide the design of neural architectures rendering the model more aligned to reasoning tasks, in a way similar to the concept of algorithmic alignment introduced by K. Xu et al. (2020).

Part II focused on factual reasoning in language-understanding tasks. As real-world knowledge graphs can comprise millions of nodes and edges, in Part I we relied on heuristics to retrieve relevant information. To address this issue, in Chapter 5 we introduced a method for reasoning over knowledge graphs that scales with the number of relation types, rather than the number of nodes or edges. We tested our approach on question answering tasks, reaching state-of-the-art results on well-known datasets.

Similarly, in Chapter 6, we focused on an entity-linking task, showing the benefit of infusing prior knowledge in the model. Concretely, we demonstrated a method to infuse prior knowledge of entity type in a simple model based on dense retrieval. Our results on standard entity linking benchmarks show that our model achieves results comparable to the performance of large generative models with 18 times more parameters.

Finally, in Part III we drove inspiration from developmental science and turned our attention to the *Core Knowledge* priors of human intelligence (Spelke et al. 2007). Concretely, we focused on geometry and topology priors and we proposed a model that incorporates these priors in the form of attention masks. We showed promising results on geometric reasoning tasks, as the model was able to learn geometric transformations from a very limited number of examples.

8.2 Future work

The work described in this thesis has many interesting avenues for further research. To start with, the method described in Chapter 7 is limited to actions on the symmetry group of the hypercubic lattice and it is not immediately extendable to other groups. For instance, though permutation matrices are still convolutions of the identity and they can be generated by a CNN, providing an architecture with predefined kernels that can compute any permutation matrix is not feasible. We believe that this can be addressed by still keeping the same overall idea of modulating attention weights using soft attention masks, possibly with a different parametrization of the masks. Recently, Otto et al. (2023) proposed a general framework to incorporate symmetry into machine learning models in three ways: enforcing known symmetries, discovering unknown symmetries and promoting symmetry during training. Though they focus on incorporating symmetry into the model, rather than learning the group action, it would be interesting to explore the connections between their framework and our definition of group-action learning in Chapter 7. We believe our formalism could provide a way to both enforce symmetry during training and discover unknown symmetries as well. Future work might focus on this research direction and on extending our work to cover a wider set of the ARC tasks.

Similarly, in Part III, we focused only on one category of *Core Knowledge* priors, namely geometry and topology priors. However, *Core Knowledge* priors further include mathematical priors, objectness priors and goal-directedness priors. Infusing these priors into neural models remains an interesting direction for future research. Concretely, we do not envision models like the one described in Chapter 7 replacing standard transformer models, but we might see them combined with transformers and other architectures in modular networks, akin to the sparse expert models proposed by Fedus et al. (2022), Zoph et al. (2022), and Du et al. (2022).

It would be interesting to extend the method of Chapter 5 to a complete set of first-order queries, including negation. We believe this could be achieved using an approach similar to the method of H. Sun, Bedrax-Weiss, et al. (2019). While H. Sun, Bedrax-Weiss, et al. (2019) rely on standard count-min sketches, we believe negation could be modelled by learned hash function to produce a compact representation of the denotation set of the query. This has two key benefits: *(i)* learned hash functions can be optimized to minimize the number of collisions, leading to improved empirical performance, and *(ii)* we can impose a regularization on the embedding space in order to learn a model that recovers the negation of a compressed answer set. This would allow supporting a complete set of first-order logical reasoning operations including negation as well.

Finally, it is important to acknowledge that in the last few years we have witnessed tremendous progress in artificial intelligence, especially driven by the introduction of large language models (OpenAI 2023; Brown et al. 2020; Radford et al. 2019). These models exhibit remarkable capabilities in a wide range of complex tasks, including mathematical problems, coding, and many others. They have also been able to show impressive reasoning abilities, but their main limitations lie in their robustness and reliability, as they “hallucinate” facts and make reasoning mistakes (OpenAI 2023). Recent work has also shown that some language models can learn simple symbolic rules but struggle with more complex ones. Even when high test accuracy seems to imply successful learning, the application of these rules is actually flawed (Kassner, Krojer, et al. 2020). Also, there is ample room for improvement especially in terms of sample efficiency and energy efficiency. In the near future, the AI community will probably focus on these challenges, rather than traditional language understanding tasks. I envision the field will at some point move towards more structured cognitive models, like the one proposed by LeCun (2022). This will hopefully address some of these limitations and allow us to move closer to the goal of building models that can learn as efficiently as humans.

Appendix

A Appendix of Chapter 2

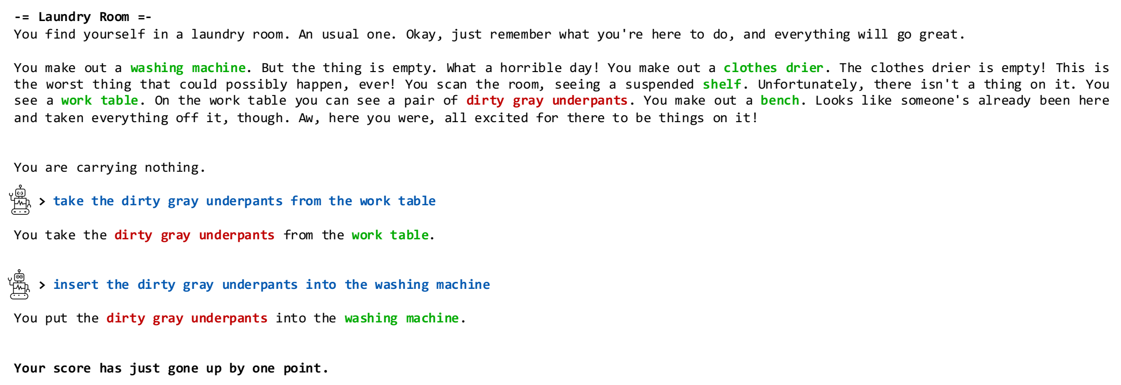
A.1 Overlap between TWC and ConceptNet

There is definitely some overlap between the resources used to build TWC and ConceptNet. However, as we discuss below, the overlap is limited and it is non-trivial for the agents to explore the knowledge graph and retrieve relevant commonsense knowledge. Indeed, only 12.2% of the *goal* entity-location pairs defined in the TWC dataset can be directly matched to a single triplet in ConceptNet. Hence, we can state that it is fair and challenging to use ConceptNet as an external source of information. At the same time, we claim that external commonsense knowledge sources can be actually useful in solving text-based games. We submit that 85.9% of the unique entities in TWC match exactly one node in ConceptNet. Moreover, 66.1% of the time, the goal location of a given entity is in its 3-hop neighborhood in ConceptNet (42.7% for a 2-hop neighborhood). This shows that an external source of commonsense like ConceptNet can help to reduce exploration while solving the games, but needs to be explored effectively. As an example, the relation between the entity `cap` and the goal location `hat_rack` can be derived from ConceptNet by following the path: `cap` $\rightarrow$ `relatedTo` $\rightarrow$ `head` $\rightarrow$ `relatedTo` $\rightarrow$ `hat` $\rightarrow$ `atLocation` $\rightarrow$ `hat_rack`.

A.2 Sample TWC games

In this section, we show and analyze an example of a TWC game instance from each difficulty level. Figures A.1, 2.2, and A.2 provide such examples together with the optimal solution to each of the analyzed games. In all figures, we highlight all objects (in red), their candidate locations (in green) and the actions taken by the agent (in blue). Note that this information is not available to the agent and is only used for illustrative purposes.

Figure A.1 shows a walk-through of an *easy* game. This game has only 1 room and 1



```
-= Laundry Room =-
You find yourself in a laundry room. An usual one. Okay, just remember what you're here to do, and everything will go great.

You make out a washing machine. But the thing is empty. What a horrible day! You make out a clothes drier. The clothes drier is empty! This is
the worst thing that could possibly happen, ever! You scan the room, seeing a suspended shelf. Unfortunately, there isn't a thing on it. You
see a work table. On the work table you can see a pair of dirty gray underpants. You make out a bench. Looks like someone's already been here
and taken everything off it, though. Aw, here you were, all excited for there to be things on it!


You are carrying nothing.

> take the dirty gray underpants from the work table

You take the dirty gray underpants from the work table.


> insert the dirty gray underpants into the washing machine

You put the dirty gray underpants into the washing machine.


Your score has just gone up by one point.
```

Figure A.1: Example of a game walkthrough belonging to the *easy* difficulty level. Best viewed in colors. Highlights are not available to the agents and are shown for illustrative purpose only.

object. We recall that this holds for all the games in the *easy* difficulty level. The game takes place in the `Laundry Room`, and the goal of the agent is to identify the correct location for the only object, in this case the `dirty gray underpants`. As all the *easy* games, the agent can reach the goal with a sequence of steps consisting of only two actions. The first action is used by the agent to take the object, and then the second action is aimed at putting the object in its goal location. In general, the goal location is not unique, but in the example shown in Figure A.1, there is only one correct location, namely the `washing machine`. Commonsense knowledge is required in the second step in order to detect the correct location among all the possible candidates.

The relatively large number of possible locations makes the *easy* games more challenging than they might look like. In our example, the locations that are not considered commonsensical for the `dirty gray underpants` are the following: `clothes drier`, `shelf`, `work table` and `bench`. Note that the `clothes drier` could have been a commonsensical location for the entity *gray underpants*, but the attribute `dirty` plays a key role. This shows that incorporating only knowledge in the form of single facts extracted from the knowledge graph is not sufficient to solve the games. On the contrary, the agent needs to aggregate commonsense knowledge from multiple triples in the knowledge graph, as previously discussed in Section A.1.

Figure 2.2 shows an example of a more complex game belonging to the *medium* difficulty level. All *medium*-level games have only 1 room and either 2 or 3 objects. The game shown in Figure 2.2 has 3 objects, namely a pair of `climbing shoes`, a `brown cap` and a `white cap`. A total of 6 steps is required to solve the game in the optimal case. These actions are reported in Figure 2.2. We can see that, similarly to what we have seen for the *easy* games, 2 steps for each object are needed. Every time that an object is placed in its goal location, the agent receives a reward, but no reward is given for the action of taking an object. Hence, the maximum final score that the agent can achieve is always equal to the number of objects, in this case 3.

Finally, Figure A.2 shows an example of the most complex games in TWC, namely the *hard* games. The game includes two rooms (`Kitchen` and `Backyard`) and the agent needs to place a total of 7 objects in the corresponding goal location. At the beginning of the game, the agent is already carrying an object (some `milk`), so it only needs to find 6 of the remaining objects. Since the game has more than 1 room, reaching the final goal may require more than 2 steps for each object. This happens because some objects may need to be carried across rooms and the in this case the agent has to visit back the initial room. In this examples, the *wet azure skirt* in the `Kitchen` has to be carried back to the the `Backyard` and placed in the `clothesline`. The provided optimal solution to the analysed game consists of the 15 actions reported in Figure A.2.

A.3 Hyperparameters and reproducibility

In addition to the hyperparameters reported in Chapter 2, we usefollowingd the experimental settings of Table A.1.

Hyperparameter	**Setting**
Batchsize	1
Hidden dimension	300
Max. # Steps	50
Discount Factor (γ)	0.9

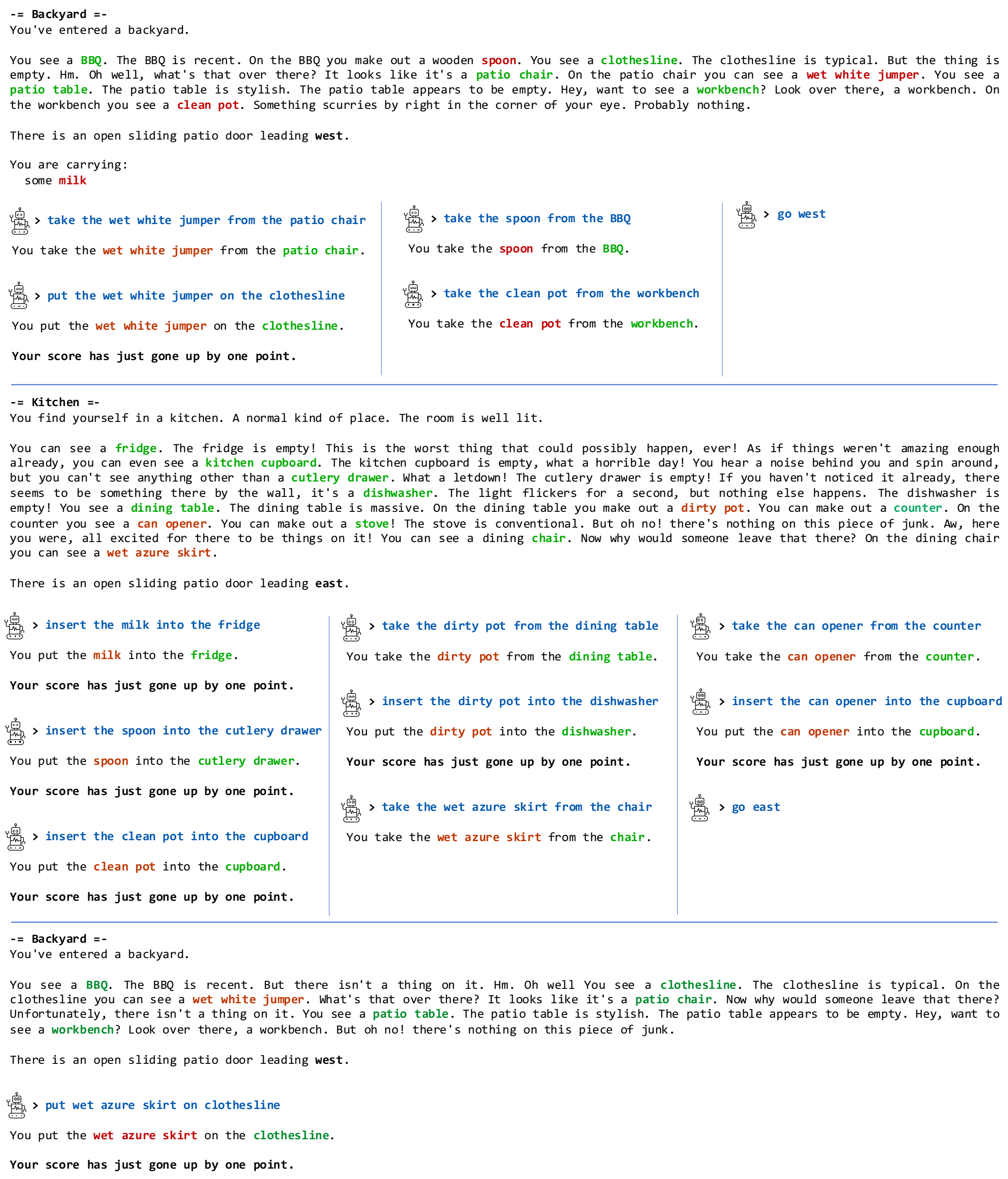
Table A.1: Hyperparameters used by the agents

The agents were trained in parallel on two machines with the specifications listed in Table A.2.

Resource	**Setting**
CPU	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz
Memory	128GB
GPUs	2 x NVIDIA Tesla V100 16 GB
Disk1	100GB
Disk2	600GB
OS	Ubuntu 18.04-64 Minimal for VSI.

Table A.2: Resources used by the agents

Each agent was trained on a single GPU for approximately 12 hours for the *Text* agent and 16 hours for the *Text + Commonsense* agent for each run.



```
-= Backyard =-
You've entered a backyard.

You see a BBQ. The BBQ is recent. On the BBQ you make out a wooden spoon. You see a clothesline. The clothesline is typical. But the thing is
empty. Hm. Oh well, what's that over there? It looks like it's a patio chair. On the patio chair you can see a wet white jumper. You see a
patio table. The patio table is stylish. The patio table appears to be empty. Hey, want to see a workbench? Look over there, a workbench. On
the workbench you see a clean pot. Something scurries by right in the corner of your eye. Probably nothing.

There is an open sliding patio door leading west.

You are carrying:
  some milk
```

```
> take the wet white jumper from the patio chair

You take the wet white jumper from the patio chair.

> put the wet white jumper on the clothesline

You put the wet white jumper on the clothesline.

Your score has just gone up by one point.

> take the spoon from the BBQ

You take the spoon from the BBQ.

> take the clean pot from the workbench

You take the clean pot from the workbench.

> go west
```

```
-= Kitchen =-
You find yourself in a kitchen. A normal kind of place. The room is well lit.

You can see a fridge. The fridge is empty! This is the worst thing that could possibly happen, ever! As if things weren't amazing enough
already, you can even see a kitchen cupboard. The kitchen cupboard is empty, what a horrible day! You hear a noise behind you and spin around,
but you can't see anything other than a cutlery drawer. What a letdown! The cutlery drawer is empty! If you haven't noticed it already, there
seems to be something there by the wall, it's a dishwasher. The light flickers for a second, but nothing else happens. The dishwasher is
empty! You see a dining table. The dining table is massive. On the dining table you make out a dirty pot. You can make out a counter. On the
counter you see a can opener. You can make out a stove! The stove is conventional. But oh no! there's nothing on this piece of junk. Aw, here
you were, all excited for there to be things on it! You can see a dining chair. Now why would someone leave that there? On the dining chair
you can see a wet azure skirt.

There is an open sliding patio door leading east.
```

```
> insert the milk into the fridge

You put the milk into the fridge.

Your score has just gone up by one point.

> insert the spoon into the cutlery drawer

You put the spoon into the cutlery drawer.

Your score has just gone up by one point.

> insert the clean pot into the cupboard

You put the clean pot into the cupboard.

Your score has just gone up by one point.

> take the dirty pot from the dining table

You take the dirty pot from the dining table.

> insert the dirty pot into the dishwasher

You put the dirty pot into the dishwasher.

Your score has just gone up by one point.

> take the wet azure skirt from the chair

You take the wet azure skirt from the chair.

> take the can opener from the counter

You take the can opener from the counter.

> insert the can opener into the cupboard

You put the can opener into the cupboard.

Your score has just gone up by one point.

> go east
```

```
-= Backyard =-
You've entered a backyard.

You see a BBQ. The BBQ is recent. But there isn't a thing on it. Hm. Oh well You see a clothesline. The clothesline is typical. On the
clothesline you can see a wet white jumper. What's that over there? It looks like it's a patio chair. Now why would someone leave that there?
Unfortunately, there isn't a thing on it. You see a patio table. The patio table is stylish. The patio table appears to be empty. Hey, want to
see a workbench? Look over there, a workbench. But oh no! there's nothing on this piece of junk.

There is an open sliding patio door leading west.

> put wet azure skirt on clothesline

You put the wet azure skirt on the clothesline.

Your score has just gone up by one point.
```

Figure A.2: Example of a game walkthrough belonging to the *hard* difficulty level. Best viewed in colors. Highlights are not available to the agents and are shown for illustrative purpose only.

B Appendix of Chapter 4

B.1 Additional results using entities as context selectors

As mentioned in Section 4.6.4, we performed an ablation study where the seeded graph attention and the state graph where not used in the retriever. Instead, we represent the context as just a single focus entity. This choice suits very well the *TWC* games, as the goal of each game is to tidy up a house by putting objects in their commonsensical locations. Hence, each rewarded action in *TWC* is of the form "*put o on s*" or "*insert o in c*", where o is an entity of type *object*, s is a *supporter*, and c is a *container* (Keerthiram Murugesan, Atzeni, Kapanipathi, Talamadupula, et al. 2021; Côté et al. 2018). As an example, a rewarded action could be "*insert dirty singlet in washing machine*", where *dirty singlet* is the *object* and *washing machine* is the container.

Representing the context as just single entities of type *object*, means that the memory of the CBR agent is storing what action to apply to each object in the game, and therefore the agent is in practice constructing a registry where each object is paired with its commonsensical location. Let c_v, c_u be two context entities. The *retriever* then computes the similarity between the contexts as:

$$\mathit{sim}(c_v, c_u) = \mathit{cosine}(\mathit{FFN}(\mathbf{h}_v), \mathit{FFN}(\mathbf{h}_u)),$$

where *cosine* denotes the cosine similarity, *FFN* is a 2-layer feed-forward network and $\mathbf{h}_v, \mathbf{h}_u$ are the BERT encodings of the `[CLS]` token for objects v and u respectively. The retriever is therefore encouraged to map objects that should be placed in the same location (either a *supporter* or a *container*) to similar representations.

Figure B.1 depicts a t-SNE (Maaten et al. 2008) visualization of the representations $\mathit{FFN}(\mathbf{h}_v)$ learned by the retriever of the **BiKE + CBR (w/o GAT)** agent. The plot shows that entities that belong to the same location are mapped to similar representations. This holds both for objects in the *in-distribution* games and for objects in the *out-of-distribution* games.

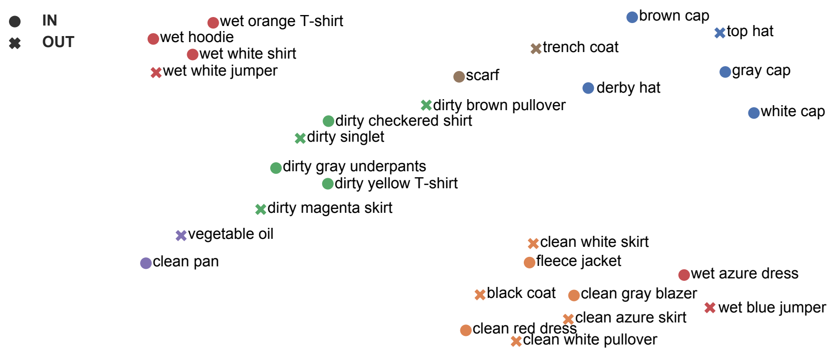


Figure B.1: Visualization of the entity representations learned by the retriever. Colors denote the target location of each object.

We evaluated all the CBR agents with the simple retriever defined above. Table B.1 reports the results for the *in-distribution* and *out-of-distribution* games. The results confirm that the entity-based context selection performs well and achieves good out-of-distribution generalization. However, we remark that the complete retriever described in Section 4.4 consistently achieves better results, showing the importance of incorporating additional structured information.

		Easy		Medium		Hard	
		#Steps	**Norm. Score**	**#Steps**	**Norm. Score**	**#Steps**	**Norm. Score**
IN	**CBR (w/o GAT)**	22.70 ± 2.05	0.81 ± 0.07	44.13 ± 1.15	0.62 ± 0.04	48.05 ± 1.30	0.33 ± 0.05
	Text + CBR (w/o GAT)	19.01 ± 3.99	0.89 ± 0.05	40.10 ± 1.52	0.67 ± 0.05	47.80 ± 1.32	0.33 ± 0.02
	TPC + CBR (w/o GAT)	17.15 ± 2.91	0.94 ± 0.04	38.32 ± 1.76	0.66 ± 0.03	47.22 ± 1.35	0.36 ± 0.03
	KG-A2C + CBR (w/o GAT)	16.67 ± 2.30	0.96 ± 0.03	38.05 ± 1.84	0.66 ± 0.04	46.45 ± 1.02	0.38 ± 0.02
	BiKE + CBR (w/o GAT)	16.32 ± 1.10	0.95 ± 0.03	36.13 ± 1.40	0.67 ± 0.04	45.72 ± 0.63	0.41 ± 0.03
OUT	**CBR (w/o GAT)**	23.90 ± 2.17	0.79 ± 0.05	44.71 ± 1.50	0.61 ± 0.04	48.87 ± 1.89	0.31 ± 0.03
	Text + CBR (w/o GAT)	21.64 ± 2.52	0.88 ± 0.02	41.12 ± 1.21	0.66 ± 0.05	48.00 ± 1.10	0.32 ± 0.06
	TPC + CBR (w/o GAT)	19.82 ± 2.13	0.92 ± 0.03	39.34 ± 1.01	0.67 ± 0.02	47.33 ± 1.30	0.36 ± 0.04
	KG-A2C + CBR (w/o GAT)	19.07 ± 2.50	0.92 ± 0.02	38.41 ± 1.94	0.65 ± 0.04	46.89 ± 2.21	0.37 ± 0.03
	BiKE + CBR (w/o GAT)	18.15 ± 1.51	0.92 ± 0.03	37.10 ± 1.41	0.67 ± 0.03	46.70 ± 0.71	0.39 ± 0.03

Table B.1: Test-set performance for *TWC in-distribution* (**IN**) and *out-of-distribution* (**OUT**) games using entities as context selectors.

B.2 Additional baselines on Jericho

Several methods have been proposed recently for text-based games. In order to keep the results in Table 4.3 more compact, we only included well-known and top-performing baselines that were evaluated on the full (or almost) set of *Jericho* games. For completeness, this section compares our best agent (**BiKE + CBR**) with the following additional methods:

- **Q*BERT** (Ammanabrolu, Tien, et al. 2020) is a deep reinforcement learning agent that plays text games by building a knowledge graph of the world and answering questions about it;

- **Trans-v-DRRN** (Y. Xu et al. 2020) relies on a lightweight transformer encoder to model the state of the game;
- **DBERT-DRRN** (Singh et al. 2021) makes use of DistilBERT (Sanh et al. 2019) fine-tuned on an independent set of human gameplay transcripts.

Table B.2 shows the scores obtained by these baselines compared to our agent enhanced with CBR. Overall, we observe that the **BiKE + CBR** agent outperforms the baselines on the majority of the games, confirming the effectiveness of case-based reasoning as a viable approach to boost the performance of text-based RL agents.

Game	Q*BERT	Trans-v-DRRN	DBERT-DRRN	BiKE + CBR
905	-	-	-	0
acorncourt	-	10	-	**12.2**
adventureland	-	25.6	-	**27.3**
afflicted	-	2	-	**3.2**
awaken	-	-	-	0
detective	246.1	288.8	-	**326.1**
dragon	-	-	-	8.3
inhumane	-	-	**32.8**	24.2
library	10.0	17	17	**22.3**
moonlit	-	-	-	0
omniquest	-	-	4.9	**17.2**
pentari	48.2	34.5	-	**52.1**
reverb	-	**10.7**	6.1	6.5
snacktime	-	-	20	**22.1**
temple	7.9	7.9	**8**	7.8
ztuu	5	4.8	-	. **87.2**
advent	-	-	-	62.1
balances	9.8	-	-	**11.9**
deephome	**1**	-	-	**1**
gold	-	-	-	2.1
jewel	-	-	**6.5**	6.4
karn	-	-	-	0
ludicorp	17.6	16	12.5	**23.8**
yomomma	-	-	0.5	**1**
zenon	-	-	-	4.1
zork1	33.6	36.4	**44.7**	44.3
zork3	-	0.19	0.2	**3.2**
anchor	-	-	-	0
enchanter	-	20.0	-	**36.3**
sorcerer	-	-	-	24.5
spellbrkr	-	40	38.2	**41.2**
spirit	-	-	2.1	**4.2**
tryst205	-	9.6	9.3	**13.4**

Table B.2: Average raw score on the *Jericho* games. Results are taken from the original papers or "–" is used if a result was not reported.

B.3 Hyperparameters and reproducibility

All CBR agents are trained using the same hyperparameter settings and the same hardware/software configuration. As mentioned in Section 4.4, we use a pre-trained

BERT model (Devlin et al. 2019) to represent initial node features in the state graph. BERT is only used to compute the initial representations of the entities and is not fine-tuned. We use the following hyperparameters for our experiments.

- We set the hidden dimensionality of the model to $d = 768$ and we use 12 attention heads for the graph attention network, each applied to 64-dimensional inputs.
- We use $n_l = 2$ seeded GAT layers for *TWC* and $n_l = 3$ for Jericho.
- On both datasets, we apply a dropout regularization on the seeded GAT with probability of 0.1 at each layer.
- Similarly, for the experiments on *TWC*, we only sample the most recent context-action pair from $\mathcal{T}$, whereas we sample $k = 3$ pairs for Jericho. We used $k = 2$ for the scalability analysis depicted in Figure 4.5.
- The retriever threshold is kept constant to $\tau = 0.7$ across all experiments.
- On *TWC*, we train the agents for 100 episodes and a maximum of 50 steps for each episode. On Jericho, as mentioned, we follow previous work and we train for $100\,000$ valid steps, starting a new episode every 100 steps or whenever the games ends.
- We set the discount factor γ to 0.9 on all experiments.
- For the ablation study on memory access, we set the output dimensionality of the RP and SRP methods to $p = 64$. For LSH, we set the number of hash tables to $l = 16$ and the length of the hash codes of $h = 8$ bits. We artificially limit the size of each bucket to the 4 most recent entries.

Experiments were parallelized on a cluster where each node was dedicated to a separate run. The configuration of the execution nodes is as reported in Table B.3.

Resource	**Setting**
CPU	Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz
Memory	128GB
GPUs	1 x NVIDIA Tesla k80 12 GB
Disk1	100GB
Disk2	600GB
OS	Ubuntu 18.04-64 Minimal for VSI

Table B.3: Hardware and software configuration used to train the agents

C Appendix of Chapter 5

C.1 Formal definition of the coalesced representation

Given a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ and a set of entities $\mathcal{V}_Q$, we can provide an alternative recursive definition of $\textit{reach}_{\mathcal{G}}(\mathcal{V}_Q, R)$ as:

$$\textit{reach}_{\mathcal{G}}(\mathcal{V}_Q, (r_1, r_2, \ldots, r_{|R|})) = \begin{cases} \mathcal{V}_Q & \text{if } |R| = 0 \\ \textit{reach}_{\mathcal{G}}(\mathcal{V}'_Q, (r_2, \ldots, r_{|R|})) & \text{if } \mathcal{V}_Q \xrightarrow{r_1} \mathcal{V}'_Q \\ \emptyset & \text{otherwise} \end{cases}$$

where $\mathcal{V}'_Q$ is the set of nodes reachable from $\mathcal{V}_Q$ by an r_1 relation.

Then, we can define the coalesced representation $\tilde{\mathcal{G}}_Q = (\tilde{\mathcal{V}}_Q, \tilde{\mathcal{R}}_Q, \tilde{\mathcal{E}}_Q)$ as follows:

- $\tilde{\mathcal{V}}_Q = \{\textit{reach}_{\mathcal{G}}(\mathcal{V}_Q, R) \mid R \in \mathcal{R}^*\}$ are the nodes R-reachable from $\mathcal{V}_Q$ by $R \in \mathcal{R}^*$, where $*$ is the Kleene star;
- $\tilde{\mathcal{R}}_Q = \mathcal{R} \cup \{\texttt{self}\}$ is the original set of relations augmented with the self-loop relation type `self`, which denotes the empty sequence $\texttt{self} \in \mathcal{R}^*$;
- edge $\mathcal{V}_i \xrightarrow{r} \mathcal{V}_j$ belongs to $\tilde{\mathcal{E}}_Q$ if and only if $\mathcal{V}_j = \textit{reach}_{\mathcal{G}}(\mathcal{V}_i, r)$, with $r \in \tilde{\mathcal{R}}_Q$.

Intuitively, this operation can be seen as coalescing relations in the original knowledge graph $\mathcal{G}$ and adding self loops. In practice, we do not need to compute all the nodes in $\tilde{\mathcal{G}}_Q$ but only edge labels.

C.2 Computational complexity

The knowledge seeking procedure described in Section 5.2.1 applies a search algorithm over the graph $\tilde{\mathcal{G}}_Q$ to obtain the most likely set of relation sequences originating from

$\mathcal{V}_Q$. The exact knowledge seeking procedure adopted in our experiments is based on the beam search algorithm and is detailed in Algorithm 2. The algorithm is designed to scale with the number of relation types in the original knowledge graph, which is usually much smaller than the number of edges (facts) or nodes (entities). In this section, we describe the algorithm in more details and we provide an extensive analysis of the computational complexity of our approach.

Overview of the knowledge seeking procedure. At each iteration, Algorithm 2 updates a set $\mathcal{B}_t$ containing triples of the form $(\mathcal{V}_t, R_t, w_t)$. We denote with $\mathcal{V}_t = \mathit{reach}_{\mathcal{G}}(\mathcal{V}_Q, R_t)$ the set of nodes reachable from $\mathcal{V}_Q$ by following R_t, whereas R_t represents a relation sequence constructed iteratively by applying the relation-level model on edges of $\tilde{\mathcal{G}}_Q$ up to time step t. The last element of the tuples w_t is the total accumulated negative log-likelihood of R_t, computed as explained in Section 5.2.1. At the beginning of the algorithm, $\mathcal{V}_1 = \mathcal{V}_Q$ is the set of entities mentioned in the natural

Algorithm 2: Knowledge Seeking

```
Input  : a coalesced knowledge graph $\tilde{\mathcal{G}}_Q$; a set of starting entities $\mathcal{V}_Q$; the beam
         width $\beta$; the maximum number of iterations $\tau_{max}$; and the number of
         relation sequences to be returned $k \leq \beta$
Output : A set of $k$ tuples of the form $(\tilde{\mathcal{A}}_Q, R, w)$, representing the $k$ most likely
         candidate answers $\tilde{\mathcal{A}}_Q$, the sequence of relations $R$ to reach $\tilde{\mathcal{A}}_Q$, and
         the negative log-likelihood $w$ of $R$

$t \leftarrow 1$
$\mathcal{B}_t \leftarrow \{(\mathcal{V}_Q, \texttt{self}, 0)\}$

repeat
    $\mathcal{B}_{t+1} \leftarrow \emptyset$
    for $(\mathcal{V}_t, R_t, w_t) \in \mathcal{B}_t$ do
        if $R_t = (r_0, \ldots, \texttt{self})$ and $t > 1$ then
            $\mathcal{B}_{t+1} \leftarrow \mathcal{B}_{t+1} \cup \{(\mathcal{V}_t, R_t, w_t)\}$
        else
            $\tilde{\mathcal{E}}_t \leftarrow \{\mathcal{V}_t \xrightarrow{r_t} \mathcal{V}_{t+1} \in \tilde{\mathcal{E}}_Q\}$
            for $\mathcal{V}_t \xrightarrow{r_t} \mathcal{V}_{t+1} \in \tilde{\mathcal{E}}_t$ do
                $R_{t+1} = (R_t, r_t)$
                $w_{t+1} \leftarrow w_t - \log \phi(\mathcal{V}_t \xrightarrow{r_t} \mathcal{V}_{t+1})$
                $\mathcal{B}_{t+1} \leftarrow \mathcal{B}_{t+1} \cup \{(\mathcal{V}_{t+1}, R_{t+1}, w_{t+1})\}$
            end
        end
    end
    $\mathcal{B}_{t+1} \leftarrow \mathsf{min}(\mathcal{B}_{t+1}, \beta)$
    $t \leftarrow t + 1$
until $\mathcal{B}_t = \mathcal{B}_{t-1}$ or $t > \tau_{max}$

return $\mathsf{min}(\mathcal{B}_t, k)$
```

language question, $R_1 = \texttt{self}$ is the empty relation sequence and we set the initial negative log-likelihood $w_1 = 0$. The algorithm receives as input a parameter β which specifies the *beam width*, namely the number of relation sequences that are expanded at each iteration. At time step t, we compute the set $\tilde{\mathcal{E}}_t$ of all edges originating from $\mathcal{V}_t$ in $\tilde{\mathcal{G}}_Q$. Then, the relation sequences R_t are expanded with the relation types labeling edges in $\tilde{\mathcal{E}}_t$. The likelihood of the new relation sequences is calculated based on w_t and the likelihood assigned by the relation-level model to the relation type appended to R_t. At the end of each iteration, the function $\mathsf{min}(\mathcal{B}_{t+1}, \beta)$ in Algorithm 2 retains for the next time step only the β tuples $(\mathcal{V}_{t+1}, R_{t+1}, w_{t+1}) \in \mathcal{B}_{t+1}$ with the minimum negative log-likelihood w_{t+1}. Note that, in Algorithm 2, relation sequences ending with the `self` relation type are not expanded after the first time step. As explained in Section 5.3, indeed, the `self` relation type is used to signal both the start and the end of the decoding process.

Time complexity. At time step t, for each triple $(\mathcal{V}_t, R_t, w_t) \in \mathcal{B}_t$, the algorithm computes ϕ for all edges $\tilde{\mathcal{E}}_t$ originating from $\mathcal{V}_t$. This means that the relation-level model described in Section 5.3 is queried $|\mathcal{B}_t| \cdot |\tilde{\mathcal{E}}_t|$ times at iteration t. Note that we do not need to compute the likelihood $\phi(\mathcal{V}_i \xrightarrow{r} \mathcal{V}_j)$ for all edges $\mathcal{V}_i \xrightarrow{r} \mathcal{V}_j$ in $\tilde{\mathcal{E}}_Q$. Let $d^+_{max}(\tilde{\mathcal{G}}_Q)$ be the maximum outdegree of nodes in $\tilde{\mathcal{G}}_Q$. At time step t, the size of the set $\mathcal{B}_{t+1}$ is restricted to β for the next iteration by the operation $\mathsf{min}(\mathcal{B}_{t+1}, \beta)$. Since $|\mathcal{B}_t|$ is bounded by β and $|\tilde{\mathcal{E}}_t|$ is bounded by $d^+_{max}(\tilde{\mathcal{G}}_Q)$, at any iteration, the relation-level model is queried at most $\beta \cdot d^+_{max}(\tilde{\mathcal{G}}_Q)$ times. Each of such queries takes constant time. The function $\mathsf{min}(\mathcal{B}_{t+1}, \beta)$ selects the β tuples in $\mathcal{B}_{t+1}$ with the smallest negative log-likelihood. This can be done on average in $\mathcal{O}(|\mathcal{B}_{t+1}|)$ time. At iteration t, the set $\mathcal{B}_{t+1}$ is initialized as the empty set and updated by adding at most $\beta \cdot d^+_{max}(\tilde{\mathcal{G}}_Q)$ tuples (one element for each query to the relation-level model). Therefore, the expected time complexity of the function $\mathsf{min}(\mathcal{B}_{t+1}, \beta)$ is $\mathcal{O}(\beta \cdot d^+_{max}(\tilde{\mathcal{G}}_Q))$. Now, note that by the definition of $\tilde{\mathcal{G}}_Q$, we have $d^+_{max}(\tilde{\mathcal{G}}_Q) \leq |\tilde{\mathcal{R}}_Q| = |\mathcal{R}| + 1$. Hence, the number of queries to the relation-level model is bounded by $\beta \cdot (|\mathcal{R}| + 1)$ and the time complexity of $\mathsf{min}(\mathcal{B}_{t+1}, \beta)$ is also $\mathcal{O}(\beta \cdot |\mathcal{R}|)$. The maximum depth reached by the knowledge seeking procedure starting from $\mathcal{V}_Q$ is bounded by τ_{max}, because Algorithm 2 performs at most τ_{max} iterations of the main outer loop. The final step $\mathsf{min}(\mathcal{B}_t, k)$ selects the k most likely tuples and can be run on average in $\mathcal{O}(\beta)$ time. This yields a final computational complexity of

$$\mathcal{O}(\tau_{max} \cdot \beta \cdot |\mathcal{R}|) = \mathcal{O}(|\mathcal{R}|).$$

Note that τ_{max} and β are constant parameters of the algorithm and are usually small. In our experiments, we set $\tau_{max} = 3$ for MetaQA 3 and $\tau_{max} = 2$ for MetaQA 2 and WebQSP. We set the beam width $\beta = 10$, obtaining only minor improvements with respect to a greedy search with $\beta = 1$. Therefore, we obtain that that time complexity of the knowledge seeking procedure scales linearly with the number of relation types and does not depend on the number of nodes or edges in $\mathcal{G}$.

Space complexity. For each iteration t, Algorithm 2 constructs $\mathcal{B}_{t+1}$ by analyzing all edges originating from each node $\mathcal{V}_t$ stored in the tuples $(\mathcal{V}_t, R_t, w_t) \in \mathcal{B}_t$. From the considerations reported above, the size of $\mathcal{B}_{t+1}$ is $\mathcal{O}(\beta \cdot |\mathcal{R}|)$. Although for notational convenience we are representing $\mathcal{B}_t$ as a set of triples, in practice we can avoid storing intermediate nodes $\mathcal{V}_t$ and construct the set of candidate answers by following R_t at the final iteration. Therefore, we only need to store relation sequences R_t and their negative log-likelihood w_t. Each tuple requires $\mathcal{O}(\tau_{max})$ space, as $|R_t|$ is bounded by τ_{max}. The space complexity of the algorithm is thus $\mathcal{O}(\tau_{max} \cdot \beta \cdot |\mathcal{R}|)$.

C.3 Expressive power

As mentioned in Section 5.2.3, the approach described in this chapter can be used to answer any valid *existential positive first order query* on a knowledge graph $\mathcal{G}$. In order to prove this, we first consider the simpler class of *conjunctive queries.* We will show a result similar to Theorem 5.2.1 for conjunctive queries, and then we will extend this result to the wider class of EPFO queries.

C.3.1 Conjunctive queries

Given a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ and a non-empty set of nodes $\mathcal{V}_Q \subseteq \mathcal{V}$, a conjunctive query on $\mathcal{G}$ is a query involving only existential quantification and conjunction operations, of the type:

$$\mathsf{Q}[\mathsf{V}_?] = \mathsf{V}_?.\exists \mathsf{V}_1, \ldots, \mathsf{V}_m : \mathsf{e}_1 \wedge \mathsf{e}_2 \wedge \cdots \wedge \mathsf{e}_{|\mathsf{Q}|},$$

such that each literal e_i is an atomic formula of the form $r(\mathsf{V}, \mathsf{V}')$, where $\mathsf{V} \in \mathcal{V}_Q \cup \{\mathsf{V}_1, \ldots, \mathsf{V}_m\}$, $\mathsf{V}' \in \{\mathsf{V}_?, \mathsf{V}_1, \ldots, \mathsf{V}_m\}$, $\mathsf{V} \neq \mathsf{V}'$, and $r(\mathsf{V}, \mathsf{V}')$ is satisfied if and only if $\mathsf{V} \xrightarrow{r} \mathsf{V}'$, $r \in \mathcal{R}$.

In general, for any query Q, we can define its *dependency graph* as the graph with nodes $\mathcal{V}_Q \cup \{\mathsf{V}_?, \mathsf{V}_1, \ldots, \mathsf{V}_m\}$. The edges of the graph are the literals $\{\mathsf{e}_1, \ldots, \mathsf{e}_{|\mathsf{Q}|}\}$, as each literal is of the form $r(\mathsf{V}, \mathsf{V}')$ and defines an edge between V and V′ (Hamilton et al. 2018). Figure C.1 shows an example of the dependency graph of a conjunctive query.

We say that a query is *valid* if its dependency graph is a directed acyclic graph (DAG), with $\mathcal{V}_Q$ as the source nodes and the target variable $\mathsf{V}_?$ as the unique sink node. In the following, we will always consider valid queries, as this ensures that the query has no redundancies nor contradictions.

Lemma C.3.1. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ be a knowledge graph and* Q *be a valid conjunctive* *query on $\mathcal{G}$. Then, there exists a sequence of relations $R^\star \in \mathcal{R}^*$ such that:*

$$\mathcal{A}_Q \subseteq \textit{reach}_\mathcal{G}(\mathcal{V}_Q, R^\star),$$

where $\mathcal{A}_Q = \{v \in \mathcal{V} \mid \mathsf{Q}[v] = \mathsf{True}\}$ is the denotation set of Q, namely the entities satisfying Q.

Proof. We proceed by induction on the number of literals $|\mathsf{Q}|$.

Base case. Assume $|\mathsf{Q}| = 1$. Then, since Q is valid, the query is of the form:

$$\mathsf{Q}[\mathsf{V}_?] = \mathsf{V}_?.r(v, \mathsf{V}_?),$$

with $\{v\} = \mathcal{V}_Q$. We have:

$$\begin{aligned}\mathcal{A}_Q &= \{v' \in \mathcal{V} \mid \mathsf{Q}[v'] = \mathsf{True}\}\\ &= \{v' \in \mathcal{V} \mid v \xrightarrow{r} v'\}\\ &= \textit{reach}_\mathcal{G}(\mathcal{V}_Q, r).\end{aligned}$$

Hence the sequence with only relation r is sufficient to generate the set of correct answers $\mathcal{A}_Q$.

Inductive step. Let Q be a conjunctive query of the form:

$$\mathsf{Q}[\mathsf{V}_?] = \mathsf{V}_?.\exists \mathsf{V}_1, \ldots, \mathsf{V}_m : \mathsf{e}_1 \wedge \mathsf{e}_2 \wedge \cdots \wedge \mathsf{e}_{|\mathsf{Q}|}.$$

Assume that there exists a sequence of relations $R^\star \in \mathcal{R}^*$, such that:

$$\mathcal{A}_Q = \{v \in \mathcal{V} \mid \mathsf{Q}[v] = \mathsf{True}\} \subseteq \textit{reach}_\mathcal{G}(\mathcal{V}_Q, R^\star).$$

Consider a query Q' constructed by adding a literal $\mathsf{e}_{|\mathsf{Q}|+1}$ to Q, and let $\mathcal{A}'_Q$ be the denotation set of Q', namely the set of nodes satisfying Q'. The conjunctive query Q' may or may not have the same target variable of Q.

If Q' shares the same target variable of Q, then Q' is of the form:

$$\mathsf{Q}'[\mathsf{V}_?] = \mathsf{V}_?.\exists \mathsf{V}_1, \ldots, \mathsf{V}_m : \mathsf{e}_1 \wedge \mathsf{e}_2 \wedge \cdots \wedge \mathsf{e}_{|\mathsf{Q}|} \wedge \mathsf{e}_{|\mathsf{Q}+1|}.$$

Note that:

$$\begin{aligned}\mathcal{A}'_Q &= \{v \in \mathcal{V} \mid \mathsf{Q}'[v] = \mathsf{True}\}\\ &\subseteq \{v \in \mathcal{V} \mid \mathsf{Q}[v] = \mathsf{True}\}\\ &\subseteq \textit{reach}_\mathcal{G}(\mathcal{V}_Q, R^\star).\end{aligned}$$

Hence, if Q and Q' share the same target variable, the same sequence of relations that generates candidate answers for Q can be used to generate candidate answers for Q'.

If Q and Q' do not have the same target variable, then we can write Q' as:

$$\mathsf{Q}'[\mathsf{V}'_?] = \mathsf{V}'_?.\exists \mathsf{V}_?, \mathsf{V}_1, \ldots, \mathsf{V}_m : \mathsf{e}_1 \wedge \mathsf{e}_2 \wedge \cdots \wedge \mathsf{e}_{|\mathsf{Q}|} \wedge \mathsf{e}_{|\mathsf{Q}+1|}.$$

Since Q' is a *valid* conjunctive query on $\mathcal{G}$, $\mathsf{e}_{|\mathsf{Q}+1|}$ is of the form $r(\mathsf{V}_?, \mathsf{V}'_?)$. Then we have:

$$\begin{aligned}
\mathcal{A}'_Q &= \{v' \in \mathcal{V} \mid \mathsf{Q}'[v'] = \mathsf{True}\} \\
&= \{v' \in \mathcal{V} \mid \exists \mathsf{V}_?, \mathsf{V}_1, \ldots, \mathsf{V}_m : \mathsf{e}_1 \wedge \mathsf{e}_2 \wedge \cdots \wedge \mathsf{e}_{|\mathsf{Q}|} \wedge \mathsf{V}_? \xrightarrow{r} v'\} \\
&= \{v' \in \mathcal{V} \mid \exists v \in \mathcal{A}_Q : v \xrightarrow{r} v'\} \\
&= \textit{reach}_{\mathcal{G}}(\mathcal{A}_Q, r) \\
&\subseteq \textit{reach}_{\mathcal{G}}(\textit{reach}_{\mathcal{G}}(\mathcal{V}_Q, R^\star), r) \\
&= \textit{reach}_{\mathcal{G}}(\mathcal{V}_Q, (R^\star, r)).
\end{aligned}$$

Therefore, the sequence $(R^\star, r)$ can be used to generate the answers to Q'. □

C.3.2 Existential Positive First-Order Queries

Any EPFO query can be expressed in disjunctive normal form (DNF), namely a disjunction of one or more conjunctions:

$$\mathsf{Q}[\mathsf{V}_?] = \mathsf{V}_?.\exists \mathsf{V}_1, \ldots, \mathsf{V}_m : \mathsf{C}_1 \vee \mathsf{C}_2 \vee \cdots \vee \mathsf{C}_{n_\vee + 1},$$

such that:

- each C_i is a conjunction of literals of the form $\mathsf{C}_i = \mathsf{e}_{i1} \wedge \mathsf{e}_{i2} \wedge \cdots \wedge \mathsf{e}_{i|\mathsf{C}_i|}$
- each literal e_{ij} is a formula of the form $r(\mathsf{V}, \mathsf{V}')$, where $\mathsf{V} \in \mathcal{V}_Q \cup \{\mathsf{V}_1, \ldots, \mathsf{V}_m\}$, $\mathsf{V}' \in \{\mathsf{V}_?, \mathsf{V}_1, \ldots, \mathsf{V}_m\}$, $\mathsf{V} \neq \mathsf{V}'$, and $r(\mathsf{V}, \mathsf{V}') = \mathsf{True}$ if and only if $\mathsf{V} \xrightarrow{r} \mathsf{V}'$, $r \in \mathcal{R}$.

As above, we assume that the Q is a *valid* query on $\mathcal{G}$, namely all C_i are valid conjunctive queries. As shown in Figure C.1, we can represent any EPFO query Q with a computation graph containing the operations that are required to answer Q. Specifically, each atomic formula can be represented as a relation projection, whereas conjunctions and disjunctions can be represented as intersection and union operations respectively.

Proof of Theorem 5.2.1 We assume that Q is expressed in disjuctive normal form and we denote with $n_\vee$ the number of disjunction ($\vee$) operators in Q. We proceed by induction on $n_\vee$.

Base case. Assume $n_\vee = 0$. Then, Q is a conjunctive query, and by Lemma C.3.1, there exists $R^\star \in \mathcal{R}^*$ such that:

$$\mathcal{A}_Q = \{v \in \mathcal{V} \mid \mathsf{Q}[v] = \mathsf{True}\} \subseteq \textit{reach}_\mathcal{G}(\mathcal{V}_Q, R^\star).$$

Inductive step. Let Q be an EPFO query in DNF:

$$\mathsf{Q}[\mathsf{V}_?] = \mathsf{V}_?.\exists \mathsf{V}_1, \ldots, \mathsf{V}_m : \mathsf{C}_1 \vee \mathsf{C}_2 \vee \cdots \vee \mathsf{C}_{n_\vee+1}.$$

Consider the subquery Q' consisting of the conjunction terms $\mathsf{C}_1 \vee \mathsf{C}_2 \vee \cdots \vee \mathsf{C}_{n_\vee}$ and assume that there exist $k \leq n_\vee$ sequences of relations $R_i^\star$ such that:

$$\mathcal{A}'_Q = \{v \in \mathcal{V} \mid \mathsf{Q}'[v] = \mathsf{True}\} \subseteq \bigcup_{i=1}^{k} \textit{reach}_\mathcal{G}(\mathcal{V}_Q, R_i^\star).$$

Note that $\mathsf{C}_{n_\vee+1}$ is a valid conjunctive query and by Lemma C.3.1 there exists $R_{k+1}^\star \in \mathcal{R}^*$ such that:

$$\{v \in \mathcal{V} \mid \mathsf{C}_{n_\vee+1}[v]\} \subseteq \textit{reach}_\mathcal{G}(\mathcal{V}_Q, R_{k+1}^\star).$$

Then, it holds that:

$$\begin{aligned}
\mathcal{A}_Q &= \{v \in \mathcal{V} \mid \mathsf{Q}[v] = \mathsf{True}\} \\
&= \{v \in \mathcal{V} \mid \mathsf{Q}'[v] \vee \mathsf{C}_{n_\vee+1}[v]\} \\
&= \mathcal{A}'_Q \cup \{v \in \mathcal{V} \mid \mathsf{C}_{n_\vee+1}[v]\} \\
&\subseteq \bigcup_{i=1}^{k} \textit{reach}_\mathcal{G}(\mathcal{V}_Q, R_i^\star) \cup \{v \in \mathcal{V} \mid \mathsf{C}_{n_\vee+1}[v]\} \\
&\subseteq \bigcup_{i=1}^{k+1} \textit{reach}_\mathcal{G}(\mathcal{V}_Q, R_i^\star).
\end{aligned}$$

□

C.4 Training strategies

In this section we describe the training strategies that we used to optimize the parameters of our relation-level model and improve generalization performance.

Supervision. For the experiments on KBQA, we assume that we only have access to pairs of questions and answers, i.e. the actual inferential chain leading from the question to the answer is latent. Therefore, we resort to weak supervision to train the model. Since at training time the set $\mathcal{A}_Q$ is known, we can compute all relation sequences $R^\star$, such that $\tilde{\mathcal{A}}_Q = \textit{reach}_\mathcal{G}(\mathcal{V}_Q, R^\star)$ is the smallest reachable superset of $\mathcal{V}_Q$.

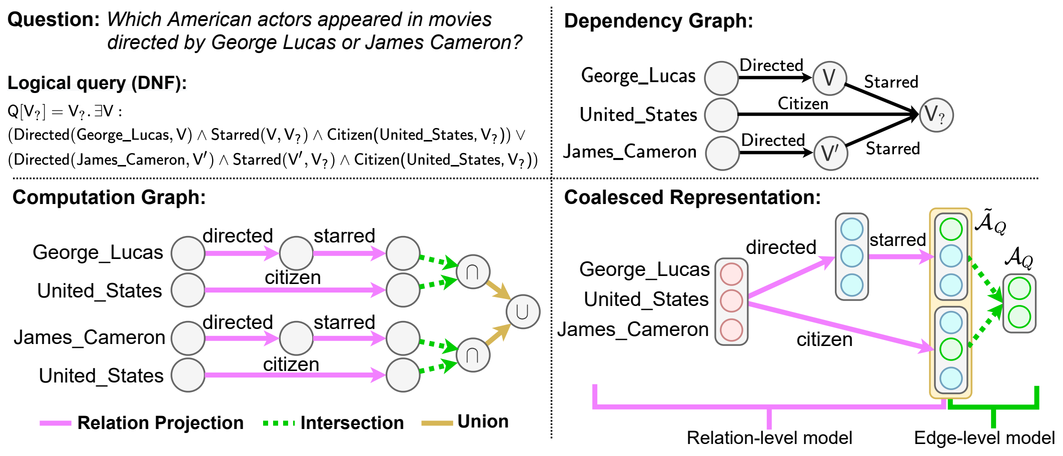


Figure C.1: Example of a natural language question and the corresponding EPFO query expressed in DNF (top left); the dependency graph of the EPFO query (top right); the computation needed to answer the query in the original KG (bottom left); and the computation performed by our approach in the coalesced representation (bottom right). Note that, for completeness, we represent two paths in the coalesced representation, but only one is sufficient.

If the smallest reachable superset of $\mathcal{A}_Q$ is not unique, all relation sequences leading to any superset of $\mathcal{A}_Q$ of minimum cardinality are considered. Note that the set of all possible relation sequences of a given length originating from $\mathcal{V}_Q$ in $\tilde{\mathcal{G}}_Q$ is much smaller than the set of all possible paths starting from nodes in $\mathcal{V}_Q$ in $\mathcal{G}$, as shown in Appendix C.5.5. Since the *CFQ* dataset contains boolean questions (where the answer is not a set of entities), for the experiment on compositional generalization we use the logical parsing provided in the dataset to compute the correct sequences of relations. We assume these sequences of relations are stored in such a way that the set of relations exiting from the a node in $\tilde{\mathcal{G}}_Q$ can be accessed efficiently in constant time. Then, at any decoding time step t, an edge is labeled as positive if and only if it belongs to a sequence of relations leading to $\tilde{\mathcal{A}}_Q$. The model is then trained using teacher forcing, namely we feed into the decoder relation sequences leading from $\mathcal{V}_Q$ to $\tilde{\mathcal{A}}_Q$. We do not have multiple decoding time steps at training time, as the whole sequence is provided at once, and relation types are appropriately masked so that they cannot attend to items in future positions.

Path dropout. Previous work (H. Sun, Dhingra, et al. 2018) has shown that randomly removing facts from the knowledge base at training time can be beneficial for generalization. Inspired by such insight, we employ a similar technique to enhance the performance of our model. Specifically, in the first epochs, we randomly remove paths that are not labeled as correct with probability p_{drop}, in order to make the problem easier for the model. This probability is the linearly decreased to 0 during training. We set the initial p_{drop} to 0.5 and we gradually lower it to 0 until half of the training epochs have been run.

Pretraining and fine tuning. For the experiments on *WebQuestionsSP* and *CFQ*, we found it beneficial to pretrain our model in order to incorporate knowledge from Freebase into the layers of the decoder. Specifically, we sampled a total of 500k 1-hop or 2-hop paths and we trained the model to predict the sequence of relations connecting two nodes, given the embeddings of the source node and the target node of the path. In order to do this, we replace the encoder with a simple 2-layer feed-forward network, with a ReLU non-linearity. This network receives as input two 100-dimensional embeddings for the source and target nodes of the path, and maps them to a d_{model}-dimensional representation. This representation is then fed into the decoder to predict the relations connecting the two nodes. We use a concatenation of 50-dimensional random and 50-dimensional pretrained TransE (Bordes et al. 2013) embeddings (Han et al. 2018) to represent the entities in the KG. Moreover, on *WebQuestionsSP* we observed that it was helpful to fine-tune BERT in order to produce better representations of the relations in the knowledge graph. The same BERT model is still used to encode both the questions and relation types.

C.5 Experimental Details

C.5.1 Datasets

KBQA datasets. We performed our experiments on KBQA on two widely adopted datasets, namely *MetaQA* (Zhang et al. 2018) and *WebQuestionsSP* (Yih et al. 2015). We provide below a detailed description of each one.

- **MetaQA**[I] (Zhang et al. 2018) is a multi-hop question answering dataset including 400K question-answer pairs. Questions are answerable using the WikiMovies knowledge base. The dataset includes 1-hop, 2-hop and 3-hop questions. It is provided under the Creative Commons Public License Attribution 3.0 Unported[II]. We evaluated our approach both on the 2-hop questions (**MetaQA 2**) and 3-hop (**MetaQA 3**) questions.
- **WebQuestionsSP** (Yih et al. 2015) comprises 4737 questions over a subset of Freebase, which is provided under the CC BY 2.5 license[III]. The questions in this dataset are answerable by performing relational following for up to two hops and an optional relational filtering operation on the result.

Table C.1 shows the number of questions in the training, development and test splits of each dataset. We use the same splits as in (H. Sun, Dhingra, et al. 2018). Table 3

[I]https://github.com/yuyuz/MetaQA
[II]https://creativecommons.org/licenses/by/3.0/
[III]https://creativecommons.org/licenses/by/2.5/

reports instead the number of triples (edges), entities (nodes) and relations in the KGs used in our experiments.

	Train	**Dev**	**Test**
MetaQA 2	118 980	14 872	14 872
MetaQA 3	114 196	14 274	14 274
WebQSP	2 848	250	1 639

Table C.1: Number of questions in the training, development and test sets

	Triples	**Entities**	**Relations**
MetaQA	392 906	43 230	18
WebQSP	23 587 078	7 448 928	575

Table C.2: Size of the knowledge graphs used for *MetaQA* and *WebQuestionsSP*

Compositional generalization. Our experiments on compositional generalization rely on the *Compositional Freebase Questions* (*CFQ*) dataset. It includes a total of 239 357 English question-answer pairs that are answerable using the public Freebase data (Bollacker et al. 2008). *CFQ* is released under the CC-BY-4.0 license[IV] provides train-test splits designed to measure the compositional generalization ability of a machine-learning model. Each question is composed of primitive elements (*atoms*), which include entity mentions, predicates and question patterns. These atoms can be combined in different ways (*compounds*) to instantiate the specific examples in the dataset. The train-test splits are designed with the twofold goal of:

1. *minimizing atom divergence*: the atoms present in the test set are also included in the training set and their distribution in the test set is as similar as possible to their distribution in the test set;
2. *maximizing compound divergence*: the distribution of compounds in the test set is as different as possible from their distribution in the training set.

The dataset provides three different splits (**MCD1**, **MCD2**, **MCD3**), with *maximum compound divergence* (MCD) and low atom divergence. For each question, both a logical parsing and the expected answers are included. Hence, *CFQ* can be used both for semantic parsing and end-to-end question answering.

C.5.2 Baselines

KBQA Baselines. On *WebQuestionsSP* and *MetaQA*, we compared our approach against the following baselines.

[IV]https://creativecommons.org/licenses/by/4.0/

- **KV-MemNN** is a key-value memory network (Miller et al. 2016) that makes use of a memory of key-value pairs to store the triples from the KG. Keys are joint representation of the subject and relation of each triple, whereas the objects of the triples are used as the corresponding values.
- **ReifKB** (Cohen, H. Sun, et al. 2020) uses a compact encoding for representing symbolic KGs, called a sparse-matrix reified KG, which can be distributed across multiple GPUs, allowing efficient symbolic reasoning.
- **GRAFT-Net** (H. Sun, Dhingra, et al. 2018) is a graph neural network designed to reason over question-specific subgraphs. The message-passing scheme is conditioned on the input question and takes inspiration from personalized page rank to perform a directed propagation of the messages starting from the entities mentioned in the question.
- **PullNet** (H. Sun, Bedrax-Weiss, et al. 2019) builds on top of GRAFT-Net and improves the quality of the question-specific subgraphs with an iterative process based on a learned classifier. This classifier selects which node should be expanded at each iteration and it is a further GNN with the same architecture as GRAFT-Net.
- **EmbedKGQA** (Saxena et al. 2020) uses KG embeddings for multi-hop question answering.
- **EmQL** (H. Sun, Arnold, et al. 2020) relies on a query embedding method that combines a count-min sketch representation for entity sets with logical operations implemented via neural retrieval over embedded KG triples.

***CFQ* Baselines.** For the experiment on compositional generalization, we compare to the best-performing baselines in *CFQ*'s public leaderboard[V]. These baselines are all designed for semantic parsing and are encoder-decoder architectures trained to output a formal query given a natural language question. Keysers et al. (2020) evaluated the compositional generalization capabilities of three sequence-to-sequence models, namely one based on LSTMs (Hochreiter et al. 1997) equipped with an attention mechanism (Bahdanau, Cho, et al. 2015) (**LSTM + Attention**), a **Transformer** (Vaswani et al. 2017) and a **Universal Transformer** (Dehghani et al. 2019). Furrer et al. (2020) conducted a study that assessed the performance of three more models. The **Evolved Transformer** (So et al. 2019) is a variation of the Transformer discovered with an evolutionary neural architecture search seeded with the original model of Vaswani et al. (2017). The *Text-to-Text Transfer Transformer* (T5) (Raffel et al. 2020) is a model pre-trained to treat every task as a text-to-text problem. Furrer et al. (2020) fine-tuned all variants, including the largest one with 11 billion parameters (**T5-11B**). Following the

[V] https://github.com/google-research/google-research/tree/master/cfq technique introduced by J. Guo et al. (2019), Furrer et al. (2020) further implemented the variant **T5-11B-mod**, which learns to predict an intermediate representation of the SPARQL query which is closer to the formulation of the questions in natural language. Finally, Y. Guo et al. (2020) introduced the *Hierarchical Poset Decoding* (**HPD**), which enforces partial permutation invariance, thus taking into account semantics and capturing higher-level compositionality.

C.5.3 Hyperparameters and reproducibility

We train the relation-level model for 300 epochs on both datasets. We use a mini-batch size of 128 for *MetaQA* and 32 for *WebQuestionsSP*. We set the dimension of the embeddings to $d_{model} = 768$, as we use 12 attention heads applied to tensors of size 64. We optimize the model using the AdamW optimizer (Loshchilov et al. 2019), with weight decay of 10^{-3}. The initial learning rate is set to 10^{-4} for *MetaQA* and $5 \cdot 10^{-6}$ for *WebQuestionsSP*. We apply dropout regularization, with probability 0.1 on both the encoder and the decoder layers. We use a beam width $\beta = 10$ for the knowledge seeking algorithm described in Appendix C.2.

BERT is fine-tuned for *WebQuestionsSP*, whereas the weights are kept fixed for *MetaQA*. For experiments on *WebQuestionsSP*, we found beneficial to pretrain our model in order to incorporate knowledge from Freebase into the layers of the decoder, as explained in Appendix C.4. Specifically, we sampled a total of 500k 1-hop or 2-hop paths and we trained the model to predict the sequence of relations connecting two nodes, given the embeddings of the source node and the target node of the path.

For the GCN-based edge-level model, we used the same implementation and hyperparameters of the version available at: https://github.com/OceanskySun/GraftNet. All experiments were performed on a NVIDIA Tesla V100 GPU with 16 GB of memory.

C.5.4 Discussion and qualitative examples

In our experiments on KBQA, for each model, we selected the entity $v^\star \in \mathcal{V}$ with the highest likelihood to be a correct answer. The answer to the question is considered correct if $v^\star \in \mathcal{A}_Q$. For the unrefined **SQALER** model, we report the expected performance selecting $v^\star$ uniformly at random from $\tilde{\mathcal{A}}_Q$.

The high performance of the unrefined model on *MetaQA* confirms our hypothesis that the relation-level model applied on the coalesced representation is sufficient for tasks such as multi-hop question answering. On *WebQuestionsSP*, the edge-level model is needed because relation projection is not sufficient to answer some of the questions in the dataset. Figure C.2 shows some examples of the relation sequences predicted by our model on the test set of *WebQuestionsSP*. For examples *(a)* and *(b)*,

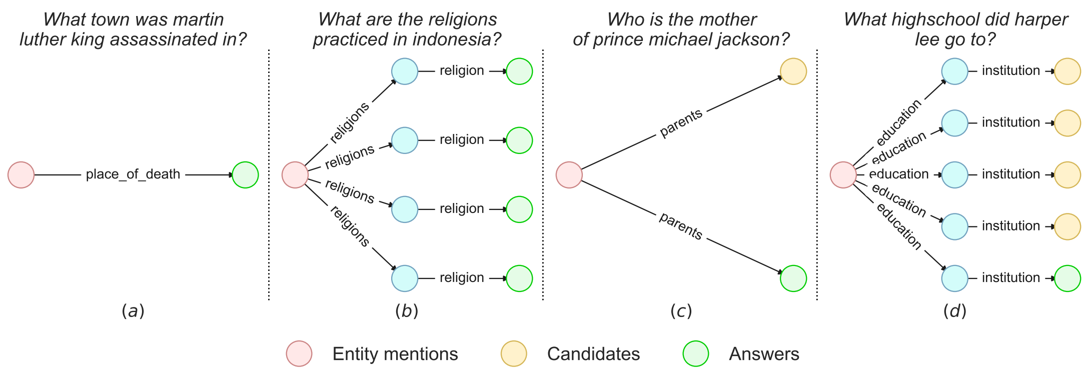

What town was martin luther king assassinated in?
place_of_death
(a)
What are the religions practiced in indonesia?
religions
religion
(b)
Who is the mother of prince michael jackson?
parents
(c)
What highschool did harper lee go to?
education
institution
(d)
Entity mentions
Candidates
Answers

Figure C.2: Example of relation sequences predicted by the unrefined relation-level model on the test set of *WebQuestionsSP*

the relation-level model is sufficient, as the set of candidates $\tilde{\mathcal{A}}_Q$ is the same as the set of the actual answers $\mathcal{A}_Q$. However, examples *(c)* and *(d)* demonstrate the need for an edge-level model, as following a sequence of relations is not always sufficient to obtain the correct answer. Note that the edge-level model is applied on a 1-hop neighborhood expansion of the graphs depicted in Figure C.2 and constrained to select an answer among the candidates $\tilde{\mathcal{A}}_Q$. Figure C.2 also shows that the answer paths for 2-hop questions in *WebQuestionsSP* always contain compound value type (CVT) entities in the middle (depicted with cyan nodes in the image). These are special entity types that are used in *Freebase* to describe n-ary relationships between entities. EmQL uses different encodings for CVT nodes and the real entities, while **SQALER** does not depend on the KG specifics.

C.5.5 Analysis of relational coalescing

In order to answer a question that requires multi-hop reasoning over a KG correctly, one should ideally either consider the full KG or a complete subgraph consisting of

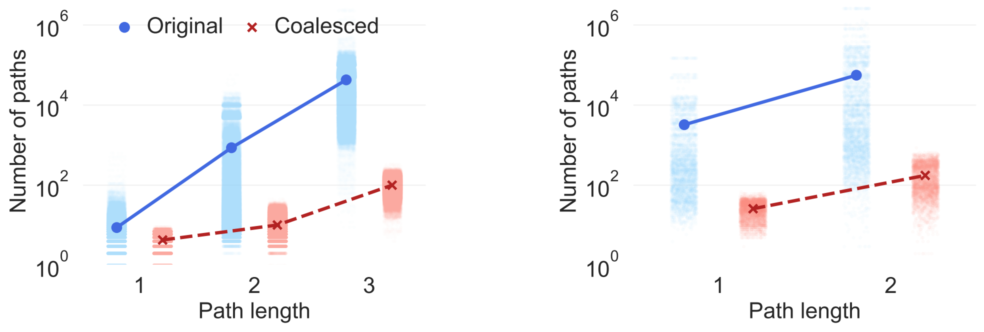


Figure C.3: Number of paths by path length in the original and coalesced graphs for *MetaQA* (right) and *WebQuestionsSP* (left)

all possible multi-hop neighbors of the entities mentioned in the question. However, such subgraphs can be very large, as shown in Table C.3. The subgraphs we analyzed include 3-hop neighbors for **MetaQA 3** and both 1 and 2-hop neighbors of the entities mentioned in the question for **WebQSP**. The average number of nodes for MetaQA 3 exceeds 10k and for the larger Freebase KG there are question subgraphs with millions of nodes. This makes impractical to perform KBQA on complete subgraphs with models that scale with the number of edges or nodes in the graph.

	MetaQA 3	**WebQSP**
Mean nodes	8.6k	36k
Max nodes	30k	1.6M
Mean facts	49k	211k
Max facts	230k	9.5M

Table C.3: Size of the subgraphs including all neighbors of the entities mentioned in a question

As a further analysis, we investigate the computational advantage of relational coalescing by computing the number of paths originating from the entities mentioned in the questions both in the original KG and in the coalesced representation. Figure C.3 presents the results for both *MetaQA* and *WebQuestionsSP*. The experiment shows that relation coalescing allows reducing the number of paths by up to 2 orders of magnitude on both datasets. This directly impacts both the memory requirements and the efficiency of our approach. For *MetaQA* we analyzed paths of length up to 3, whereas for *WebQuestionsSP* we consider paths of length 1 or 2, as the dataset does not include 3-hop questions.

D Appendix of Chapter 6

D.1 Duck typing on knowledge graphs

To get more insights into our definition of duck typing on knowledge graphs, we performed a qualitative analysis of entities that share a large number of relations in Wikidata (Vrandečić et al. 2014). Precisely, we used the cardinality of the symmetric difference between the sets of relations of two entities $e_1, e_2 \in \mathcal{E}$, defined as:

$$\begin{aligned} \mathit{dist}_{\mathit{KG}}(e_1, e_2) &= |\mathcal{R}(e_1) \triangle \mathcal{R}(e_2)| \\ &= |(\mathcal{R}(e_1) \setminus \mathcal{R}(e_2)) \cup (\mathcal{R}(e_2) \setminus \mathcal{R}(e_1))| \\ &= |(\mathcal{R}(e_1) \cup \mathcal{R}(e_2)) \setminus (\mathcal{R}(e_1) \cap \mathcal{R}(e_2))| \end{aligned}$$

as a measure of the distance between the types of two entities e_1 and e_2. Notice that the distance defined above can be expressed as the Hamming distance between binary encodings of the sets of relations, hence we can efficiently retrieve the neighbors of a given entity on GPU, following the method of Jeff Johnson et al. (2021). If our definition of duck typing works well, we expect entities with low distance to be likely of the same type. Table D.1 shows the top-10 neighbors that minimize the distance function defined above for several entities. We emphasize that these lists of neighbors are not produced by our model, rather they are examples of the prior knowledge that we aimed to infuse in DUCK. This analysis shows that our notion of duck typing carries fine-grained type information, as it allows detecting countries, cities, highly influential computer scientists and mathematicians, football players, singers, politicians, animals, companies, scientific awards and more.

D.2 Entity-box distance

This section provides more details on the entity-box distance function defined in Section 6.3.2. A plot of the distance function in the uni-dimensional case, for a scalar

Italy	**London**	**Rome**	**Alan Turing**	**Ada Lovelace**
Portugal	Istanbul	Milan	Bernhard Riemann	Lady Byron
Spain	Madrid	Florence	Kurt Gödel	Rosalind Franklin
Norway	Istanbul Province	Naples	John von Neumann	Elizabeth Fry
Greece	Stockholm	Venice	Herbert A. Simon	Catherine Dickens
Poland	Cairo	Turin	John Forbes Nash Jr.	Rosina Bulwer Lytton
Denmark	Buenos Aires	Palermo	Claude Shannon	Lady Emmeline Stuart-Wortley
Belgium	Manchester	Rio de Janeiro	Nikolai Lobachevsky	Eleanor Marx
Hungary	Amsterdam	Genoa	Benoit Mandelbrot	Wilhelmina Powlett, Duchess of Cleveland
Finland	Milan	Bologna	Willard Van Orman Quine	Rachel Russell, Lady Russell
Republic of Ireland	Ankara	Lisbon	Niels Henrik Abel	Martha Jefferson
Cristiano Ronaldo	**Justin Bieber**	**Donald Trump**	**Lion**	**Jaguar**
Lionel Messi	Harry Styles	Joe Biden	Tiger	Cougar
Luis Suárez	Chris Brown	Mrs. Bill Clinton	Leopard	Ocelot
Gerard Piqué	Ed Sheeran	Barack Obama's	Cheetah	Giant anteater
Neymar	Eminem	George W. Bush	Jaguar	Giant armadillo
Manuel Neuer	Camila Cabello	Al Gore	Red panda	Chimpanzee
Paul Pogba	Nick Jonas	Kamala Harris	Giant panda	Bonobo
Ronaldinho	Jordin Sparks	Elizabeth Warren	Cougar	Indian rhinoceros
Luka Modrić	Richard Marx	Bill Clinton	Okapi	Giant otter
Antoine Griezmann	Niall Horan	Michael Bloomberg	Hippopotamus	Black rhinoceros
Gareth Bale	Shawn Mendes	Benjamin Netanyahu	Fennec fox	Pronghorn
Jaguar Cars	**Maserati**	**Veliko Tarnovo**	**Gracilinanus**	**Fields Medal**
Land Rover	Lancia	Kluczbork	Scolomys	IEEE Medal of Honor
Steyr-Daimler-Puch	VinFast	Yambol	Aethalops	Kavli Prize
MG Cars	McLaren Automotive	Kyustendil	Oligoryzomys	Rosenstiel Award
Gulf Oil	Massimo Dutti	Targovishte	Raphicerus	Paul Ehrlich and Ludwig Darmstaedter Prize
British Motor Corporation	chapter Mate	Pančevo	Vesper mouse	Albert Einstein World Award of Science
Safeway (UK)	Infiniti	Kragujevac	Rhabdomys	Canada Gairdner International Award
Rover Group	Peroni Brewery	Ruse, Bulgaria	Balantiopteryx	Earle K. Plyler Prize for Molecular Spectroscopy
Rover Company	Lotus Cars	Sombor	Arielulus	Dannie Heineman Prize for Mathematical Physics
MIPS Technologies	Colruyt (supermarket)	Vratsa	Bassariscus	Bôcher Memorial Prize
F. W. Woolworth Company	Overkill Software	Pazardzhik	Microsciurus	NAS Award in Chemical Sciences

Table D.1: Top 10 entities with the most similar set of relations to a given entity in Wikidata

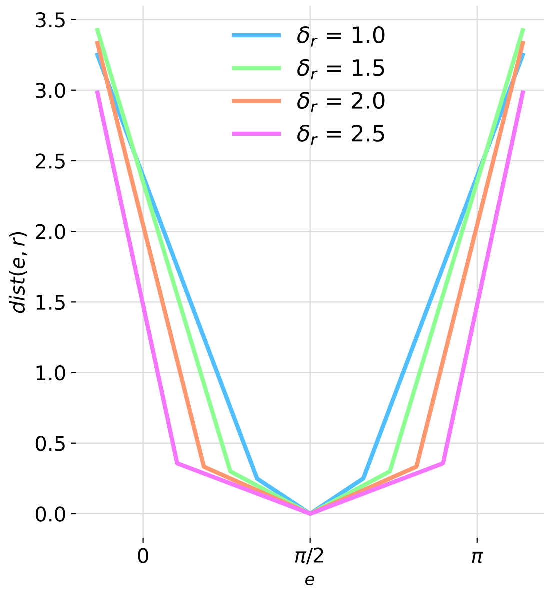


Figure D.1: Plot of the entity-box distance in the uni-dimensional case, for boxes centered at $\pi/2$ with different scalar widths δ_r

entity representation e and several boxes centered at $\pi/2$ with different scalar widths δ_r is shown in Figure D.1. The plot shows that the distance function has different slopes for entities inside and outside the boxes. This is meant to strongly penalize entities that lie outside boxes corresponding to their relations, as it ensures that outside points receive high gradient through which they can more easily reach their target box. Additionally, recalling the expression for *dist*(e, r) given in Section 6.3.2, notice that the distance depends on the width of the box. More precisely, whenever an entity is inside its target box, the distance inversely correlates with box size. This allows maintaining low distance values inside large boxes while providing a gradient to keep points inside. For entities outside their target boxes, the distance linearly correlates with the width of the box, to penalize points outside larger boxes more severely.

D.3 Details on the model

In order to train DUCK, we need to convert the entity representations $\boldsymbol{e}$ into spherical polar coordinates (the same applies to the mention representations $\boldsymbol{m}$). This can be done as follows:

$$
\begin{aligned}
\varphi_{e,1} &= \arccos\left(\frac{e_1}{\sqrt{e_d^2 + e_{d-1}^2 + \cdots + e_1^2}}\right)\\
\varphi_{e,2} &= \arccos\left(\frac{e_2}{\sqrt{e_d^2 + e_{d-1}^2 + \cdots + e_2^2}}\right)\\
&\vdots\\
\varphi_{e,d-2} &= \arccos\left(\frac{e_{d-2}}{\sqrt{e_d^2 + e_{d-1}^2 + e_{d-2}^2}}\right)\\
\varphi_{e,d-1} &= \begin{cases} \arccos\left(\frac{e_{d-1}}{\sqrt{e_d^2+e_{d-1}^2}}\right) & \text{if } e_d \geq 0\\ 2\pi - \arccos\left(\frac{e_{d-1}}{\sqrt{e_d^2+e_{d-1}^2}}\right) & \text{if } e_d < 0 \end{cases}
\end{aligned}
$$

where e_i is the i-th entry of the entity representation $\boldsymbol{e} \in \mathbb{R}^d$ and $\varphi_{e,i}$ is the i-th component of the representation in spherical coordinates $\boldsymbol{\varphi}_e \in \mathbb{R}^{d-1}$. Looking at the equations above, we notice that in a spherical coordinate systems, all angles range from 0 to π, with the only exception of the last coordinate $\varphi_{e,d-1}$, which ranges from 0 to π if e_d is positive and from π to 2π otherwise. In order to make the definition of the boxes and of the entity-box distance simpler, we decided to constrain the last coordinate in the range $[0, \pi]$ as well. We achieved this objective by constraining the last coordinate of the entity and mention representations to be positive, applying an absolute value to the last dimension of the output of the entity and mention encoders. This essentially restricts all representations and boxes to be on half of the hypersphere, more precisely on the portion where $e_d > 0$. We apply this transformation before computing the overall optimization objective of Section 6.3.2.

D.4 Training details

To train DUCK, we need to select negative entities e_j for the entity-disambiguation loss $\mathcal{L}_{ED}$ of Section 6.3. Having high-quality negative entities is crucial to achieve high performance, thus we trained DUCK in several stages.

First, we trained the model using, as negative entities for each mention, all entities in the same batch. In order to provide more meaningful information, we further added entities that maximize a prior probability $\hat{p}(e|m)$, extracted from count statistics derived from large text corpora. In details, we used the prior probabilities of Ayoola et al. (2022). Notice that, differently from Ayoola et al. (2022), we do not use these prior probabilities at inference time, but we only use them to provide better negative entities to the model in this first training stage. For each mention, we included in a batch 3 negative entities that maximize the prior probability, limiting the total number of entities in a batch to 32 (per GPU). In this stage, we use a batch size of 16 mentions (per GPU), which means that, for some mentions, we do not have the entities that maximize the prior probability. This is not an issue, as we still use all entities in the same batch as negatives for every mention. To compute the loss $\mathcal{L}_{Duck}$, we used a sampling temperature $\alpha = 0.1$. Furthermore, in order to provide a better training signal and counteract missing information in the knowledge graph, we only trained the model using entities that have at least 5 relations.

We trained the model for 1 epoch on 8 GPUs, validating on the BLINK validation set every 5000 gradient steps. Then, we used the model that maximizes the validation performance to produce a representation for every entity, and we mined the closest representations for each entity in Wikipedia. This step is usually referred to as hard-negative mining. We used these hard negatives to train the model again, starting from the same checkpoint used for the negative mining. We used a batch size of 16, with 3 negative examples for each mention and up to 32 entities in a batch. We increased the sampling temperature for the boxes to $\alpha = 0.5$, keeping a threshold of at least 5 relations for each entity. We trained the model for one more epoch, validating every 5000 gradient steps as before.

Finally, we repeated the hard-negative mining process and kept training the model for $10\,000$ additional gradient steps, using a batch size of 4, 5 hard negatives for each mention and up to 3 entities that maximize the prior probability $\hat{p}(e|m)$ (if distinct from the negatives). We did gradient accumulation for 4 steps, increased the maximum length of a mention from 128 tokens to 512, and set the sampling temperature to $\alpha = 1.0$. In this final stage, we assumed the model had already learned to place entities

Member of political party	Member of sports team	Cast member
Richard Nixon	Justin Moore (soccer)	A Time to Sing (film)
Édouard Philippe	Scott Jones (Puerto Rican footballer)	A Secret Life (film)
Kaname Tajima	Blake Camp	The Dream (1989 film)
Laurent Fabius	Miles Robinson (soccer)	Can You Hear the Laughter? The Story of Freddie Prinze
Albert II, Prince of Monaco	Simon Thomas (soccer)	I Was a Teenage TV Terrorist
Joe Biden	Scott Wilson (footballer, born 1993)	A Sinful Life
François Hollande	Scott Jenkins (soccer)	The Morning After (1986 film)
Jean-Marc Ayrault	Ali Mohamed (footballer)	Cries Unheard: The Donna Yaklich Story
Ursula von der Leyen	Scott Fraser (footballer, born 1995)	My Sex Life... or How I Got into an Argument
Yasutomo Suzuki	Justin Willis (soccer)	Enemies, A Love Story (film)

Table D.2: Closest Wikipedia entities to different boxes according to the entity-box distance function. Correct predictions are highlighted in green, whereas predictions that do not match relations in Wikidata are highlighted in red. Best viewed in color.

Member of political party	Member of sports team	Cast member
Ronald Reagan	Tayfun Korkut	The Crying Game
Jacques Chirac	Lilian Thuram	Ice Cold in Alex
George H. W. Bush	Scott Sanders (baseball)	Michael Collins (film)
Madeleine Albright	Roberto Carlos (footballer)	Viva Zapata!
Deng Xiaoping	Tony Adams (footballer)	On the Waterfront
Bob Dole	Stuart McCall	Lawrence of Arabia (film)
Masoud Barzani	Joakim Persson	My Turn (memoir)
Yasser Arafat	Cosmin Contra	Aidan Quinn
Bill Clinton	Todd Martin	Der Spiegel
Liam Neeson	Geoff Aunger	Julia Roberts

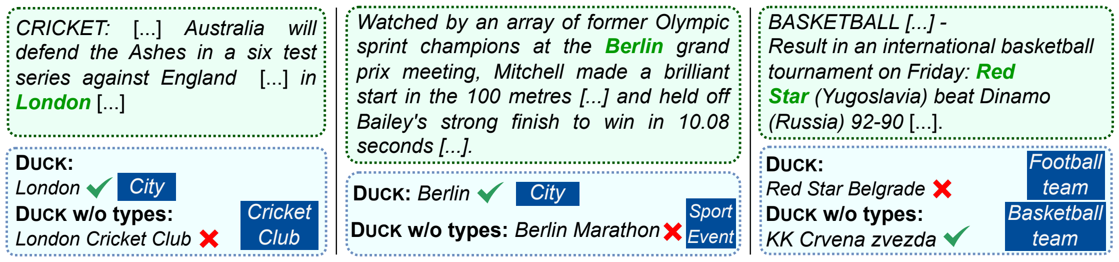
Table D.3: Closest entities (extracted from the validation set of the **AIDA** dataset) to different boxes according to the entity-box distance function. Correct predictions are highlighted in green, whereas predictions that do not match relations in Wikidata are highlighted in red. Only 6 entities in **AIDA** have the relation *Cast member* and the model is able to correctly retrieve all of them, has shown above. Best viewed in color.

in their target boxes, hence we used all entities in the dataset, regardless of the number of relations they have in Wikidata.

D.5 Additional qualitative results

Following the qualitative analyses of Section 6.5.4, in this section we provide additional results and further examples.

Analysis of the boxes. Table D.2 shows additional examples of the top-10 entities lying closer to the center of a box. This analysis is performed on all entities in Wikipedia (approximately 6M entities for the English language) and complements the examples reported on the left side of Table 6.4. In this case, we analyzed three more relations, namely *member of political party*, *member of sports team*, and *cast member*. The model correctly reports politicians for the first box, athletes for the second, and movies for the latter, confirming the clustering of entity types that we noticed in Section 6.5.4. Additionally, the model appears robust to missing information in the knowledge graph,



CRICKET: [...] Australia will defend the Ashes in a six test series against England [...] in **London** [...]

DUCK: London ✓ City

DUCK w/o types: London Cricket Club ✗ Cricket Club

Watched by an array of former Olympic sprint champions at the **Berlin** grand prix meeting, Mitchell made a brilliant start in the 100 metres [...] and held off Bailey's strong finish to win in 10.08 seconds [...].

DUCK: Berlin ✓ City

DUCK w/o types: Berlin Marathon ✗ Sport Event

BASKETBALL [...] - Result in an international basketball tournament on Friday: **Red Star** (Yugoslavia) beat Dinamo (Russia) 92-90 [...].

DUCK: Red Star Belgrade ✗ Football team

DUCK w/o types: KK Crvena zvezda ✓ Basketball team

Figure D.2: Further examples of the predictions of **DUCK** and **DUCK w/o types**. Mentions highlighted in **bold green**.

being able to predict the relation *cast member* for movies that are missing it in the KG.

We performed the same analysis, using the same set of relation, on the entities appearing in the validation set of the **AIDA** dataset. The results are reported in Table D.3. Since **AIDA** contains news articles, the dataset includes several mentions of politicians and athletes, and the model is able to correctly cluster the two types of entities (with only one error in the top 10 predictions for the relation *member of political party*). On the other hand, the dataset includes only 6 entities that are movies (more precisely, entities with the relation *cast member*). Interestingly, the top-10 entities closer to the center of the box corresponding to the relation *cast member* are all the movies mentioned in AIDA. The remaining 4 entities listed in Table D.3 include two actors (*Aidan Quinn* and *Julia Roberts*), suggesting that the embedding space carries semantic information and that actors are closer to movies than other entities.

Examples. Figure D.2 shows further examples of the predictions of DUCK and the ablation **DUCK w/o types**. Confirming the insights of Figure 6.2, the first two examples (left and center), show that DUCK is usually able to predict entities of the correct type and how this can help the model in making the correct prediction. The third example (right) shows a case where the model predicts a wrong entity, as it links the mention to a football team, though the context clearly suggests that the correct entity should be a basketball team instead. This suggests that, in some rare cases, DUCK might give too much weight to the prior knowledge about the relations of candidate entities, loosing knowledge coming from the description of the entity and from contextual information about the mention.

D.6 Hyperparameters and reproducibility

We trained DUCK using the AdamW optimizer (Loshchilov et al. 2019) on 8 NVIDIA A100 GPUs, each with 40 GB of memory. Following Ayoola et al. (2022), we initialized the learning rate to 0 and linearly increased it up to 1.00×10^{-5} over the first 5000 steps. To avoid catastrophic forgetting, we set a maximum learning rate of 1.00×10^{-6} when fine-tuning on **AIDA**. We limited the number of entities in a batch (which are used to compute the loss term $\mathcal{L}_{ED}$) to 32 per GPU, but we shared entity representations across all devices when computing the loss, reaching an effective maximum number of entities of $32 \times 8 = 256$ per batch. We increased the maximum length of a mention to 512 tokens at inference time. Table D.4 reports the values of all hyperparameters of the model for reproducibility of our results.

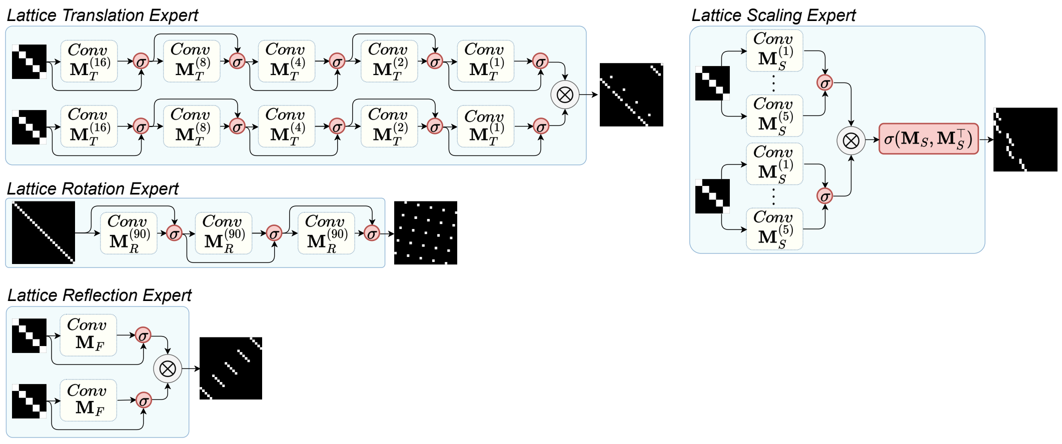
Hyperparameter	**Value**
Learning rate (max)	1.00×10^{-5}
Learning rate warm-up steps	5000
γ	2
λ_{Duck}	0.1
λ_{l_2}	0.1
Number of negative boxes k	512
$\delta_{\min}$	0.1
$\delta'_{\min}$	0.1
d	1024
α	See Appendix D.4
Max entity length n_e	128
Max mention length n_m	See Appendix D.4
Max relation length n_r	256
Batch size	See Appendix D.4
Max num. entities per batch (per GPU)	32

Table D.4: Hyperparameter values of DUCK

E Appendix of Chapter 7

E.1 Additional details on the model

This section describes the LatFormer architecture providing additional details that were not covered in Section 7.4.1. As mentioned in Section 7.4.1, it is possible to design convolutional neural networks that perform all considered transformations of the lattice. Figure E.1 shows the architecture of the four expert models that generate *translation*, *rotation*, *reflection* and *scaling* masks.



Lattice Translation Expert
Conv $\mathbf{M}_T^{(16)}$
Conv $\mathbf{M}_T^{(8)}$
Conv $\mathbf{M}_T^{(4)}$
Conv $\mathbf{M}_T^{(2)}$
Conv $\mathbf{M}_T^{(1)}$
σ
$\otimes$
Lattice Scaling Expert
Conv $\mathbf{M}_S^{(1)}$
Conv $\mathbf{M}_S^{(5)}$
$\sigma(\mathbf{M}_S, \mathbf{M}_S^{\top})$
Lattice Rotation Expert
Conv $\mathbf{M}_R^{(90)}$
Lattice Reflection Expert
Conv $\mathbf{M}_F$

Figure E.1: Model architecture of all the mask experts that we considered.

All models are CNNs applied to the identity matrix. In the figure, we use the following notation:

- $M_T^{(\delta)}$ denotes an attention mask implementing a translation by δ along one dimension;
- $M_R^{(90)}$ denotes an attention mask implementing a translation by $90°$;
- M_F denotes an attention mask implementing a reflection along one dimension;

- $\boldsymbol{M}_S^{(h)}$ denotes an attention mask implementing an upscaling by h along one dimension.

Using Corollary 7.3.3, we can derive the kernels of the convolutional layers shown in Figure E.1. These kernels are frozen at training time, the model only learns the gating function, denoted as σ in the figure. Notice that all the models follow the same overall structure. However, for scaling, we also learn an additional gate, denoted as $\sigma(\boldsymbol{M}_S, \boldsymbol{M}_S^\top)$ in the Figure E.1. This gate allows the model to transpose the mask and serves the purpose of implementing down-scaling operations (down-scaling is the transpose of up-scaling).

The composition of more actions can be obtained by combining different experts. This can be done either by chaining the experts or by matrix multiplication of the masks. In preliminary experiments, we did not notice any significant difference in performance between the two options and we rely on the latter in our implementation.

E.2 Additional experiments and details on the setup

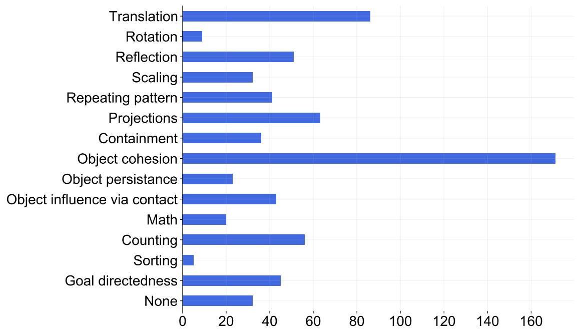
This section provides additional details on the experimental setup of all our experiments, including further information on the generation of the synthetic tasks and the data annotation process for ARC.

E.2.1 Experiments on synthetic data

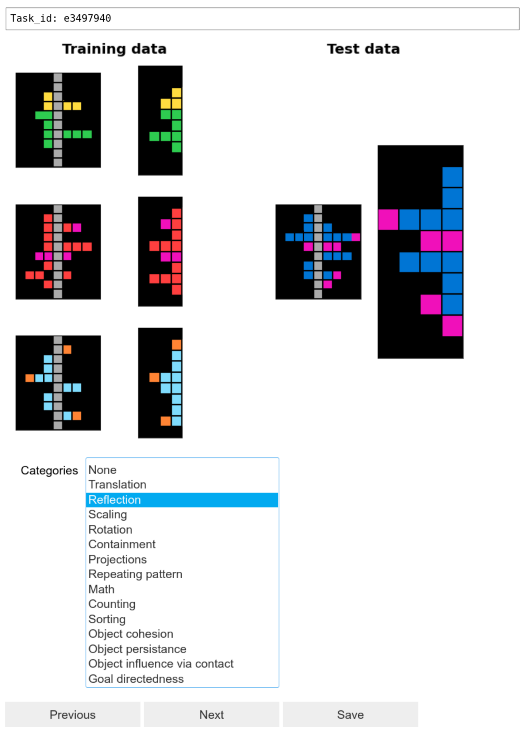
We considered four categories of tasks, namely *translation*, *rotation*, *reflection* and *scaling*. Each task is defined in terms of input-output pairs, which are sampled from the set of all ARC grids and padded to the size of 30×30 cells. To each input grid, a synthetic transformation is applied in order to obtain the corresponding output grid. For each task in each category, we generated 2048 training pairs and 100 test pairs.

For translation tasks, we have a total of 900 possible translations in a 30×30 grid. However, generating data and training models on 900 tasks is computationally expensive, so we randomly sampled 5 translations in the interval $[1, 29] \times [1, 29]$, obtaining a total of 100 translation tasks. Rotation tasks include all 4-fold rotations except the identity. Similarly, reflection tasks involve horizontal, vertical and diagonal reflections. Scaling tasks include all possible up/down scaling transformations of the input grid by factors of $[2, 5] \times [2, 5]$ for a total of 32 scaling tasks.

The models are evaluated based on the mean accuracy on each category. For each task we compute the accuracy on the test set based on how many of the predicted images match exactly the ground truth.



(a)



(b)

Figure E.2: Distribution of the considered core knowledge priors across the ARC tasks (a) and user interface built to annotate the dataset (b).

E.2.2 Experiments on ARC

In order to experiment with ARC, we first performed an annotation of the dataset to identify the underlying knowledge priors for each task. To this end, we built a user interface where the annotator could browse the tasks and label them by selecting any combination of the available knowledge priors. Figure E.2b shows the user interface provided to the annotator, whereas Figure E.2a shows the distribution of knowledge priors across the ARC tasks. Most tasks follow in more than one of the categories represented in Figure E.2a.

ARC can be regarded as a meta-learning benchmark, as it provides a set of training tasks and a set of unseen tasks to evaluate the performance of the model learned on the meta-training data. It is important to emphasize that we do not target this use case, as we instead use the same setup as in the synthetic data and learn each task from scratch using only its training set.

Though simple and elegant, the supervised-learning formulation prevents our models from reusing knowledge that can be shared between different tasks. In order to mitigate this issue, we rely on a data-augmentation strategy. At training time, for each model and every iteration, we augment each grid 10 times by mapping each color to a different color (using the same mapping across training examples). The rationale behind this data-augmentation strategy is that *(1)* we assume that for tasks involving only geometric knowledge priors to be not affected by color mapping and *(2)* all models (including LATFORMER) need to learn a function from d-dimensional color representations to categorical variables, hence it is beneficial if all colors are represented in the training set.

All models are evaluated based on the ratio of solved tasks and a task is considered solved if the model can predict the correct output grid for *all* examples in the test set.

E.2.3 Experiments on LARC

All baselines relying on program synthesis for the experiment on LARC are taken from the work of Acquaviva et al. (2021). They share an underlying formulation based on the *generate-and-check* strategy. The program synthesizer generates a program *prog* given a natural program *natprog* (which can defined by either the input-output pairs alone or by input-output pairs and the corresponding natural language description) from the following distribution:

$$P_{synth}(prog \mid natprog) \propto P_{gen}(prog \mid natprog)\mathbb{1}[prog \vdash IO].$$

Above, P_{gen} is the generative distribution and $\mathbb{1}[prog \vdash IO]$ is the checker. The generative distribution proposes programs by first generating a tree bigram over the grammar of a DSL and then enumerating deterministically programs from a probabilistic context free grammar fitted to this bigram distribution in decreasing probability. For simplicity, Acquaviva et al. (2021) used an unconditioned generator $P_{synth}(prog)$ (i.e., a fitted prior) when language is absent, and language-conditioned models $P_{synth}(prog \mid NL)$ when a natural language description *NL* is given.

Once a program has been proposed, the checker validates the program *prog* by executing it on the interpreter, ensuring that $prog(x) = y$ for all input-output pairs $(x, y) \in IO$, where *IO* denotes the set of input-output pairs. The key strength of this approach lies in its generalizability, as programs that can be checked successfully on all training examples are likely to generalize.

E.2.4 Additional experiments on image registration

As an additional experiment, to assess the applicability of our LATFORMER on natural images, we performed experiments on multimodal *image registration*, namely the problem of spatially aligning images from different modalities. Image registration is a well-studied problem in computer vision and we do not aim to establish state-of-the-art performance. The main purpose of this experiment is giving a hint on

	Aerial data			Cytological data		
	α-AMD	SIFT	LatFormer	α-AMD	SIFT	LatFormer
CycleGAN $(A \to B)$	5.3 ± 3.1	67.2 ± 16.8	**68.3 ± 4.5**	**74.2 ± 3.8**	30.2 ± 4.2	68.3 ± 2.2
CycleGAN $(B \to A)$	65.7 ± 6.7	84.0 ± 2.5	**86.1 ± 3.1**	21.3 ± 1.8	18.2 ± 3.5	**24.2 ± 3.3**
DRIT++ $(A \to B)$	35.3 ± 2.4	38.1 ± 8.1	**38.2 ± 5.9**	50.4 ± 12.1	24.2 ± 2.7	**62.7 ± 10.2**
DRIT++ $(B \to A)$	20.2 ± 2.1	38.3 ± 4.5	**43.2 ± 4.1**	**30.1 ± 4.5**	5.2 ± 3.1	15.6 ± 3.5
pixel2pixel $(A \to B)$	84.2 ± 4.0	**98.7 ± 0.4**	89.3 ± 2.2	53.2 ± 6.9	9.5 ± 1.0	**61.2 ± 5.5**
pixel2pixel $(B \to A)$	68.2 ± 7.5	87.5 ± 4.03	**89.7 ± 3.3**	0.2 ± 0.1	4.0 ± 1.0	**4.2 ± 1.1**
StarGAN $(A \to B)$	63.1 ± 7.8	7.4 ± 2.7	**72.2 ± 6.3**	**60.2 ± 12.2**	12.2 ± 2.0	59.5 ± 5.9
StarGAN $(B \to A)$	52.1 ± 4.0	7.9 ± 1.3	**53.3 ± 4.0**	**20.8 ± 3.9**	4.1 ± 0.9	13.4 ± 3.1
CoMIR	94.2 ± 5.7	**100.0 ± 0.0**	90.2 ± 3.3	76.2 ± 12.1	74.1 ± 6.3	**78.1 ± 3.4**

Table E.1: Results of the experiment on image registration. The rows represent different models trained to translate images from modality A to B $(A \to B)$ or viceversa $(B \to A)$.

the applicability of our method to natural images beyond ARC. We refer the reader to SuperGlue (Sarlin et al. 2020) and COTR (Jiang et al. 2021) to have a sense of approaches specifically designed for this task.

Popular approaches to multimodal image registrations work in two stages: first, they learn a model that converts one modality into the other (or to transfer both modalities in the same representation as proposed by Pielawski et al. (2020)), then they align the two images using traditional techniques. We follow the experimental setup of Lu et al. (2021) and experiment with two datasets, one containing aerial views of a urban neighborhood and one containing cytological images. The images we employ are views of the same scene, but they are taken with different modalities and they are translated with respect to one another. We use the code of the authors to generate data involving only translations. Lu et al. (2021) additionally consider small rotations, but these transformations are not actions in the symmetry group of a lattice, so we are not interested in resolving them.

We employ several state-of-the-art methods for modality translation and we compare our method to α-AMD (Lindblad et al. 2014) and SIFT (Lowe 1999) based on the success rate metric defined by Lu et al. (2021). A registration is considered successful if the relative registration error (i.e., the residual distance between the reference patch and the transformed patch after registration normalized by the height and width of the patch) is below 2%. Table E.1 reports our results on the image registration tasks and shows that our approach performs well on both datasets coupled with different methods for modality translation. We use the same models of Lu et al. (2021) for the modality translation task. Then, in order to solve the image registration task, we divide each image into 30×30 patches and we run our model to predict the translation from one patch in an image to its counterpart in the corresponding image.

E.3 Deferred Proofs

We prove both Theorem 7.3.1 and 7.3.2 by induction on the dimensionality of the hypercubic lattice m.

E.3.1 Base case for theorems 1 and 2

First, it is useful to notice that whenever $\boldsymbol{M} \in \{0,1\}^{n\times n}$ has exactly a single 1 per row, in other words $\boldsymbol{M} \cdot \mathbf{1}_n = \mathbf{1}_n$, then, for any $\boldsymbol{X} \in \mathbb{R}^{n\times d}$

$$\begin{aligned}
\operatorname{MaskedAttention}(\boldsymbol{X}; \boldsymbol{M}) &= \frac{\boldsymbol{A}}{\boldsymbol{A} \cdot \mathbf{1}_n \mathbf{1}_n^\top} \boldsymbol{X} \\
&= \frac{\operatorname{softmax}\left(\frac{\boldsymbol{X}\boldsymbol{X}^\top}{\sqrt{d}}\right) \odot \boldsymbol{M}}{\operatorname{softmax}\left(\frac{\boldsymbol{X}\boldsymbol{X}^\top}{\sqrt{d}}\right) \odot \boldsymbol{M} \cdot \mathbf{1}_n \mathbf{1}_n^\top} \boldsymbol{X} \\
&= \boldsymbol{M} \cdot \boldsymbol{X}.
\end{aligned}$$

In order to prove the theorem, we need to show that, for any action $\mathrm{g} \in \mathsf{G}_1$, including translations, reflections and rotations, there exists a mask $\boldsymbol{M}_{\mathrm{g}}$ such that:

$$\operatorname{MaskedAttention}(\boldsymbol{X}; \boldsymbol{M}_{\mathrm{g}}) = \mathrm{g} \circ \boldsymbol{X}.$$

Let us consider different families of actions separately. **Translation.** As mentioned in Section 7.3.2, in the 1-dimensional case, a translation by one element to the right for a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^\top$ is given by the circulant permutation matrix:

$$\boldsymbol{M} = \boldsymbol{M}_T^{(1)} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}.$$

This holds because $\boldsymbol{M}_T^{(1)} \cdot \mathbf{1}_n = \mathbf{1}_n$, so:

$$\operatorname{MaskedAttention}(\boldsymbol{x}; \boldsymbol{M}_T^{(1)}) = \boldsymbol{M}_T^{(1)} \cdot \boldsymbol{x} = (x_n, x_1, x_2, \ldots, x_{n-1})^\top.$$

In general, a translation by δ elements is given by the circulant matrix $\boldsymbol{M}_T^{(\delta)} = (\boldsymbol{M}_T^{(1)})^\delta$. Therefore, masks implementing translation operations exist in the 1-dimensional case and they are circulant permutation matrices. This is enough for a base case for

Theorem 7.3.1.

For Theorem 7.3.2, simply notice that:

$$\boldsymbol{M}_T^{(\delta)} = \mathcal{F}^{-1}\Big(\mathcal{F}(\boldsymbol{I}_n)\, \exp(-\frac{2\pi j}{n}\, \boldsymbol{o}_T^{(\delta)}\, \boldsymbol{r}_n^\top)\Big) \quad \text{where} \quad \boldsymbol{o}_T^{(\delta)} = \begin{pmatrix} -\delta \\ -\delta \\ \vdots \\ -\delta \end{pmatrix}.$$

This comes directly from the time-shifting property of the Fourier transform.

Reflection. In the 1-dimensional case, the reflection of a vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_n)^\top$ is:

$$\text{MaskedAttention}(\boldsymbol{x}; \boldsymbol{M}_F) = \boldsymbol{M}_F \cdot \boldsymbol{x} = (x_n, x_{n-1}, \ldots, x_2, x_1)^\top$$

with

$$\boldsymbol{M}_F = \begin{pmatrix} 0 & \cdots & 0 & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 1 & 0 & 0 \\ \vdots & \cdots & \vdots & \vdots & \vdots \\ 1 & \cdots & 0 & 0 & 0 \end{pmatrix}.$$

The attention mask $\boldsymbol{M}_F$ can be obtained by shifting the rows of the identity matrix by:

$$\boldsymbol{o}_F = \begin{pmatrix} n-1 \\ n-3 \\ n-5 \\ \vdots \\ 1 \end{pmatrix}.$$

Therefore, by the time-shifting property of the Fourier transform we have:

$$\boldsymbol{M}_F = \mathcal{F}^{-1}\Big(\mathcal{F}(\boldsymbol{I}_n)\, \exp(-\frac{2\pi j}{n}\, \boldsymbol{o}_F\, \boldsymbol{r}_n^\top)\Big).$$

Rotation. Rotation (4-fold) is not defined in one dimension, so for a base case we need to consider the square lattice. Let $\boldsymbol{X} \in \mathbb{R}^{l_1 \cdot l_2}$ be a vectorized representaiton of a $n = l_1 \times l_2$ dimensional matrix. We need to define a vector $\boldsymbol{o}_R \in \mathbb{R}^n$ such that:

$$\boldsymbol{M}_R^{(90)} = \mathcal{F}^{-1}\Big(\mathcal{F}(\boldsymbol{I}_n)\, \exp(-\frac{2\pi j}{n}\, \boldsymbol{o}_R\, \boldsymbol{r}_n^\top)\Big)$$

is a rotation mask. Since rotation is a permutation of the identity, we know the vector exists. As $\boldsymbol{X}$ is vectorized, the $\boldsymbol{o}_R^{(90)}$ needs to take into account the size of the first dimension l_1. For example, in order to perform a rotation on a vectorized representation, we need to map the first element of $\boldsymbol{X}$ to the position $(l_1 - 1)$. The reader can check that the vector given by

$$(\boldsymbol{o}_R^{(90)})_k = k \cdot (l_1 - 1) - \lfloor (k-1)/l_1 \rfloor$$

satisfies the equation above.

Scaling. Although scaling is not a group action of the symmetry group of the lattice, we pointed out that it still can be defined within the same general formulation as the other transformations. We can take the 1-dimensional lattice as a base case and consider a vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_n)^\top$. Let $h \in \mathbb{N}$ be a parameter specifying the filter size of the scaling operation. As an example, for $h = 2$ we have:

$$\mathrm{MaskedAttention}(\boldsymbol{x}; \boldsymbol{M}_S^{(h)}) = \boldsymbol{M}_S^{(h)} \cdot \boldsymbol{x} = (x_1, x_1, x_2, x_2, \ldots, x_{\lfloor n/2 \rfloor})^\top,$$

where:

$$\boldsymbol{M}_S^{(h)} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

This kind of matrix can also be obtained by shifting the rows of the identity as follows:

$$\boldsymbol{M}_S^{(h)} = \mathcal{F}^{-1}\Big(\mathcal{F}(\boldsymbol{I}_n) \exp(-\frac{2\pi j}{n} \, \boldsymbol{o}_S^{(h)} \, \boldsymbol{r}_n^\top)\Big),$$

where $(\boldsymbol{o}_S^{(h)})_k = (k - 1 \bmod h) + (h-1) \cdot \lfloor (k-1)/h \rfloor$.

E.3.2 Inductive step for theorems 1 and 2

Suppose that $\boldsymbol{M}_{\mathrm{g}_1} \in \{0,1\}^{n_1 \times n_1}$ and $\boldsymbol{M}_{\mathrm{g}_2} \in \{0,1\}^{n_2 \times n_2}$ are attention masks implementing actions $\mathrm{g}_1 \in \mathsf{G}_{m_1}$ and $\mathrm{g}_2 \in \mathsf{G}_{m_2}$ on some tensors $\mathsf{X}_1 \in \mathbb{R}^{l_1 \times \cdots \times l_{m_1}}$ and $\mathsf{X}_2 \in \mathbb{R}^{l'_1 \times \cdots \times l'_{m_2}}$, with $n_1 = l_1 \cdot \ldots \cdot l_{m_1}$ and $n_2 = l'_1 \cdot \ldots \cdot l'_{m_2}$. Consider a tensor $\mathsf{X} \in \mathbb{R}^{l_1 \times \cdots \times l_{m_1} \times l'_1 \times \cdots \times l'_{m_2}}$ and its vectorization $\boldsymbol{X} \in \mathbb{R}^n$ with $n = n_1 n_2$.

We have:

$$\begin{aligned} &\mathrm{MaskedAttention}(\boldsymbol{X}; \boldsymbol{M}_{\mathrm{g}_1} \otimes \boldsymbol{M}_{\mathrm{g}_2}) = \\ &\quad = (\boldsymbol{M}_{\mathrm{g}_1} \otimes \boldsymbol{M}_{\mathrm{g}_2}) \, \boldsymbol{X} \\ &\quad = (\boldsymbol{M}_{\mathrm{g}_1} \otimes \boldsymbol{I}_{n_2})(\boldsymbol{I}_{n_1} \otimes \boldsymbol{M}_{\mathrm{g}_2}) \boldsymbol{X} \\ &\quad = \mathrm{MaskedAttention}(\mathrm{MaskedAttention}(\boldsymbol{X}; \boldsymbol{I}_{n_1} \otimes \boldsymbol{M}_{\mathrm{g}_2}); \boldsymbol{M}_{\mathrm{g}_1} \otimes \boldsymbol{I}_{n_2}). \end{aligned}$$

Now notice that:

$$\begin{aligned}\mathrm{MaskedAttention}(\boldsymbol{X}; \boldsymbol{I}_{n_1} \otimes \boldsymbol{M}_{\mathrm{g}_2}) &= (\boldsymbol{I}_{n_1} \otimes \boldsymbol{M}_{\mathrm{g}_2})\, \boldsymbol{X} \\ &= \mathrm{vec}(\boldsymbol{M}_{\mathrm{g}_2}\, \mathrm{vec}^{-1}(\boldsymbol{X})\, \boldsymbol{I}_{n_1}) \\ &= \mathrm{vec}(\boldsymbol{M}_{\mathrm{g}_2}\, \mathrm{vec}^{-1}(\boldsymbol{X})),\end{aligned}$$

and similarly

$$\begin{aligned}\mathrm{MaskedAttention}(\boldsymbol{X}; \boldsymbol{M}_{\mathrm{g}_1} \otimes \boldsymbol{I}_{n_2}) &= (\boldsymbol{M}_{\mathrm{g}_1} \otimes \boldsymbol{I}_{n_2})\, \boldsymbol{X} \\ &= \mathrm{vec}(\boldsymbol{I}_{n_2}\, \mathrm{vec}^{-1}(\boldsymbol{X})\, \boldsymbol{M}_{\mathrm{g}_1}^\top) \\ &= \mathrm{vec}((\boldsymbol{M}_{\mathrm{g}_1}\, \mathrm{vec}^{-1}(\boldsymbol{X})^\top)^\top).\end{aligned}$$

Therefore, we conclude that performing masked attention with the mask $\boldsymbol{M}_{\mathrm{g}_1} \otimes \boldsymbol{M}_{\mathrm{g}_2}$ on X is equivalent to applying g_1 on the first m_1 dimensions and g_2 on the last m_2 dimensions of X. This provides a way for building attention masks for higher-dimensional lattices using the primitive masks defined in Section E.3.1, proving both Theorem 7.3.1 and 7.3.2.

E.3.3 Proof of Corollary 1

The proof of Corollary 1 follows immediately from Theorem 7.3.2 and from the property of the Fourier transform according to which multiplying in the Fourier domain implements a convolution in the original domain.

Bibliography

Aamodt, Agnar and Enric Plaza (1994). "Case-based reasoning: Foundational issues, methodological variations, and system approaches". In: *AI communications* 7.1, pp. 39–59.

Abboud, Ralph, İsmail İlkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori (2020). "BoxE: A Box Embedding Model for Knowledge Base Completion". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.* Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. URL: https://proceedings.neurips.cc/paper/2020/hash/6dbbe6abe5f14af882ff977fc3f35501-Abstract.html.

Acquaviva, Samuel, Yewen Pu, Marta Kryven, Catherine Wong, Gabrielle E. Ecanow, Maxwell I. Nye, Theodoros Sechopoulos, Michael Henry Tessler, and Joshua B. Tenenbaum (2021). "Communicating Natural Programs to Humans and Machines". In: *CoRR* abs/2106.07824. arXiv: 2106.07824. URL: https://arxiv.org/abs/2106.07824.

Adhikari, Ashutosh, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and William L Hamilton (2020). "Learning Dynamic Knowledge Graphs to Generalize on Text-Based Games". In: *arXiv preprint arXiv:2002.09127.*

Adolphs, Leonard and Thomas Hofmann (2019). "LeDeepChef: Deep Reinforcement Learning Agent for Families of Text-Based Games". In: *ArXiv* abs/1909.01646.

Aghajanyan, Armen, Bernie Huang, Candace Ross, Vladimir Karpukhin, Hu Xu, Naman Goyal, Dmytro Okhonko, Mandar Joshi, Gargi Ghosh, Mike Lewis, and Luke Zettlemoyer (2022). "CM3: A Causal Masked Multimodal Model of the Internet". In: *CoRR* abs/2201.07520. arXiv: 2201.07520. URL: https://arxiv.org/abs/2201.07520.

Ammanabrolu, Prithviraj and Matthew J. Hausknecht (2020). "Graph Constrained Reinforcement Learning for Natural Language Action Spaces". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview.net. URL: https://openreview.net/forum?id=B1x6w0EtwH.

Ammanabrolu, Prithviraj and Mark Riedl (2019). "Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers),* pp. 3557–3565.

Ammanabrolu, Prithviraj, Ethan Tien, Matthew J. Hausknecht, and Mark O. Riedl (2020). "How to Avoid Being Eaten by a Grue: Structured Exploration Strategies for Textual Worlds". In: *CoRR* abs/2006.07409. arXiv: 2006.07409. URL: https://arxiv.org/abs/2006.07409.

Andreoli, Jean-Marc (2019). "Convolution, attention and structure embedding". In: *arXiv preprint arXiv:1905.01289*.

Angeli, Gabor, Melvin Jose Johnson Premkumar, and Christopher D. Manning (July 2015). "Leveraging Linguistic Structure For Open Domain Information Extraction". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 344–354. DOI: 10.3115/v1/P15-1034. URL: https://aclanthology.org/P15-1034.

Atzeni, Mattia and Maurizio Atzori (2018a). "Towards Semantic Approaches for General-Purpose End-User Development". In: *Second IEEE International Conference on Robotic Computing, IRC 2018, Laguna Hills, CA, USA, January 31 - February 2, 2018*. IEEE Computer Society, pp. 369–376. DOI: 10.1109/IRC.2018.00077. URL: https://doi.org/10.1109/IRC.2018.00077.

– (2018b). "Translating Natural Language to Code: An Unsupervised Ontology-Based Approach". In: *First IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2018, Laguna Hills, CA, USA, September 26-28, 2018*. IEEE Computer Society, pp. 1–8. DOI: 10.1109/AIKE.2018.00009. URL: https://doi.org/10.1109/AIKE.2018.00009.

– (2018c). "What Is the Cube Root of 27? Question Answering Over CodeOntology". In: *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference*. Vol. 11136. Lecture Notes in Computer Science. Springer, pp. 285–300.

Atzeni, Mattia, Jasmina Bogojeska, and Andreas Loukas (2021). "SQALER: Scaling Question Answering by Decoupling Multi-Hop and Logical Reasoning". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. URL: https://openreview.net/forum?id=2CQQ_C1i0b.

Atzeni, Mattia, Shehzaad Zuzar Dhuliawala, Keerthiram Murugesan, and Mrinmaya Sachan (2022). "Case-based reasoning for better generalization in textual reinforcement learning". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. URL: https://openreview.net/forum?id=ZDaSIkWT-AP.

Atzeni, Mattia, Mikhail Plekhanov, Frédéric A. Dreyer, Nora Kassner, Simone Merello, Louis Martin, and Nicola Cancedda (2023). "Polar Ducks and Where to Find Them: Enhancing Entity Linking with Duck Typing and Polar Box Embeddings". In: *The 2023 Conference on Empirical Methods in Natural Language Processing*. URL: https://openreview.net/forum?id=RVQccn8rcr.

Atzeni, Mattia, Mrinmaya Sachan, and Andreas Loukas (2023). "Infusing Lattice Symmetry Priors in Attention Mechanisms for Sample-Efficient Abstract Geometric Reasoning". In: *International Conference on Machine Learning, ICML 2023, 23-29 July*

2023, Honolulu, Hawaii, USA. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 1200–1217. URL: https://proceedings.mlr.press/v202/atzeni23a.html.

Ayoola, Tom, Shubhi Tyagi, Joseph Fisher, Christos Christodoulopoulos, and Andrea Pierleoni (2022). "ReFinED: An efficient zero-shot-capable approach to end-to-end entity linking". In: *NAACL 2022.*

Ba, Lei Jimmy, Jamie R. Kiros, and Geoffrey E. Hinton (2016). "Layer Normalization". In: *CoRR* abs/1607.06450. arXiv: 1607.06450. URL: http://arxiv.org/abs/1607.06450.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1409.0473.

Bahdanau, Dzmitry, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron C. Courville (2019). "Systematic Generalization: What Is Required and Can It Be Learned?" In: *ICLR 2019.*

Ballard, Dana H. (2000). *An introduction to natural computation.* Complex adaptive systems. MIT Press. ISBN: 978-0-262-52258-8.

Basu, Kinjal, Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Kartik Talamadupula, Tim Klinger, Murray Campbell, Mrinmaya Sachan, and Gopal Gupta (2022). "A Hybrid Neuro-Symbolic approach for Text-Based Games using Inductive Logic Programming". In: *Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations.* URL: https://openreview.net/forum?id=NzjyY2Z9zJd.

Batson, Joshua D., Daniel A. Spielman, Nikhil Srivastava, and Shang-Hua Teng (2013). "Spectral sparsification of graphs: theory and algorithms". In: *Commun. ACM* 56.8, pp. 87–94. DOI: 10.1145/2492007.2492029. URL: https://doi.org/10.1145/2492007.2492029.

Battaglia, Peter W., Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu (2018). "Relational inductive biases, deep learning, and graph networks". In: *CoRR* abs/1806.01261. arXiv: 1806.01261.

Bengio, Yoshua (2017). "The Consciousness Prior". In: *CoRR* abs/1709.08568. arXiv: 1709.08568.

– (2019). *Deep Learning for System 2 Processing.* Invited talk at AAAI 2019, http://www.iro.umontreal.ca/~bengioy/AAAI-9feb2020.pdf.

Bengio, Yoshua, Tristan Deleu, Nasim Rahaman, Nan R. Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher J. Pal (2020). “A Meta-Transfer Objective for Learning to Disentangle Causal Mechanisms”. In: *ICLR 2020*.

Bengio, Yoshua, Nicholas Léonard, and Aaron C. Courville (2013). “Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation”. In: *CoRR* abs/1308.3432. arXiv: 1308.3432. URL: http://arxiv.org/abs/1308.3432.

Bianchi, Reinaldo AC, Luiz A Celiberto Jr, Paulo E Santos, Jackson P Matsuura, and Ramon Lopez de Mantaras (2015). “Transferring knowledge as heuristics in reinforcement learning: A case-based approach”. In: *Artificial Intelligence* 226, pp. 102–121.

Bianchi, Reinaldo AC, Paulo E Santos, Isaac J Da Silva, Luiz A Celiberto, and Ramon Lopez de Mantaras (2018). “Heuristically accelerated reinforcement learning by means of case-based reasoning and transfer learning”. In: *Journal of Intelligent & Robotic Systems* 91.2, pp. 301–312.

Bollacker, Kurt, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor (2008). “Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge”. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’08. Association for Computing Machinery, pp. 1247–1250.

Booch, Grady, Francesco Fabiano, Lior Horesh, Kiran Kate, Jonathan Lenchner, Nick Linck, Andrea Loreggia, Keerthiram Murugesan, Nicholas Mattei, Francesca Rossi, and Biplav Srivastava (2021). “Thinking Fast and Slow in AI”. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, pp. 15042–15046. DOI: 10.1609/AAAI.V35I17.17765. URL: https://doi.org/10.1609/aaai.v35i17.17765.

Bordes, Antoine, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko (2013). “Translating Embeddings for Modeling Multi-relational Data”. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, pp. 2787–2795.

Bosselut, Antoine, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi (2019). “COMET: Commonsense Transformers for Automatic Knowledge Graph Construction”. In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, pp. 4762–4779. DOI: 10.18653/v1/p19-1470. URL: https://doi.org/10.18653/v1/p19-1470.

Botha, Jan A., Zifei Shan, and Daniel Gillick (2020). “Entity Linking in 100 Languages”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Association for Computational Linguistics, pp. 7833–7845. DOI: 10.18653/v1/2020.emnlp-main.630. URL: https://doi.org/10.18653/v1/2020.emnlp-main.630.

Branavan, SRK, David Silver, and Regina Barzilay (2012). “Learning to win by reading manuals in a monte-carlo framework”. In: *Journal of Artificial Intelligence Research* 43, pp. 661–704.

Broeck, Guy Van den (2019). *Computers and Thought Award Lecture at the International Joint Conference on Artificial Intelligence 2019.*

Bronstein, Michael M, Joan Bruna, Taco Cohen, and Petar Veličković (2021). “Geometric deep learning: Grids, groups, graphs, geodesics, and gauges”. In: *arXiv preprint arXiv:2104.13478.*

Brown, Tom B. et al. (2020). “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.* Ed. by Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Celiberto Jr, Luiz A, Jackson P Matsuura, Ramon Lopez De Mantaras, and Reinaldo AC Bianchi (2011). “Using cases as heuristics in reinforcement learning: a transfer learning application”. In: *Twenty-Second International Joint Conference on Artificial Intelligence.*

Chen, Qian, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei (2018). “Neural natural language inference models enhanced with external knowledge”. In: *Proceedings of ACL 2018.*

Chen, Shuang, Jinpeng Wang, Feng Jiang, and Chin-Yew Lin (2020). “Improving Entity Linking by Modeling Latent Entity Type Information”. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020.* AAAI Press, pp. 7529–7537. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6251.

Chen, Ting, Martin R. Min, and Yizhou Sun (2018). “Learning K-way D-dimensional Discrete Codes for Compact Embedding Representations”. In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018.* Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 853–862. URL: http://proceedings.mlr.press/v80/chen18g.html.

Cho, Kyunghyun, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *EMNLP 2014.*

Chollet, François (2019). “On the Measure of Intelligence”. In: *CoRR* abs/1911.01547. arXiv: 1911.01547. URL: http://arxiv.org/abs/1911.01547.

Cohen, William W., Haitian Sun, R. Alex Hofer, and Matthew Siegler (2020). “Scalable Neural Methods for Reasoning With a Symbolic Knowledge Base”. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview.net. URL: https://openreview.net/forum?id=BJlguT4YPr.

Cohen, William W., Fan Yang, and Kathryn Mazaitis (2017). “TensorLog: Deep Learning Meets Probabilistic DBs”. In: *CoRR* abs/1707.05390. arXiv: 1707.05390. URL: http://arxiv.org/abs/1707.05390.

Cordonnier, Jean-Baptiste and Andreas Loukas (2019). “Extrapolating Paths with Graph Neural Networks”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019.* Ed. by Sarit Kraus. ijcai.org, pp. 2187–2194. DOI: 10.24963/ijcai.2019/303. URL: https://doi.org/10.24963/ijcai.2019/303.

Cordonnier, Jean-Baptiste, Andreas Loukas, and Martin Jaggi (2020). “On the Relationship between Self-Attention and Convolutional Layers”. In: *Eighth International Conference on Learning Representations-ICLR 2020.* CONF.

Côté, Marc-Alexandre, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler (2018). “TextWorld: A Learning Environment for Text-based Games”. In: *CoRR* abs/1806.11532.

Cropper, Andrew and Stephen H. Muggleton (2019). “Learning efficient logic programs”. In: *Mach. Learn.* 108.7, pp. 1063–1083. DOI: 10.1007/S10994-018-5712-6. URL: https://doi.org/10.1007/s10994-018-5712-6.

Csordas, Robert and Juergen Schmidhuber (2019). “Improving Differentiable Neural Computers Through Memory Masking, De-allocation, and Link Distribution Sharpness Control”. In: *ICLR 2019.*

Cucerzan, Silviu (2007). “Large-Scale Named Entity Disambiguation Based on Wikipedia Data”. In: *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic.* Ed. by Jason Eisner. ACL, pp. 708–716. URL: https://aclanthology.org/D07-1074/.

Cui, Leyang, Sijie Cheng, Yu Wu, and Yue Zhang (2021). “On Commonsense Cues in BERT for Solving Commonsense Tasks”. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 683–693.

Dalvi, Nilesh N. and Dan Suciu (2012). “The dichotomy of probabilistic inference for unions of conjunctive queries”. In: *J. ACM* 59.6, 30:1–30:87. DOI: 10.1145/2395116.2395119. URL: https://doi.org/10.1145/2395116.2395119.

Das, Rajarshi, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum (2018). “Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning”. In: *ICLR 2018.*

Das, Rajarshi, Ameya Godbole, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum (2020). “A simple approach to case-based reasoning in knowledge bases”. In: *arXiv preprint arXiv:2006.14198*.

Das, Rajarshi, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum (2018). “Building Dynamic Knowledge Graphs from Text using Machine Reading Comprehension”. In: *CoRR* abs/1810.05682. arXiv: 1810.05682. URL: http://arxiv.org/abs/1810.05682.

Das, Rajarshi, Arvind Neelakantan, David Belanger, and Andrew McCallum (2017). “Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Volume 1: Long Papers*. Association for Computational Linguistics, pp. 132–141.

Das, Rajarshi, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay-Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum (2021). “Case-based Reasoning for Natural Language Queries over Knowledge Bases”. In: *arXiv preprint arXiv:2104.08762*.

Dasgupta, Shib Sankar, Michael Boratko, Dongxu Zhang, Luke Vilnis, Xiang Li, and Andrew McCallum (2020). “Improving Local Identifiability in Probabilistic Box Embeddings”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. URL: https://proceedings.neurips.cc/paper/2020/hash/01c9d2c5b3ff5cbba349ec39a570b5e3-Abstract.html.

Davey, B. A. and H. A. Priestley (2002). *Introduction to Lattices and Order*. 2nd ed. Cambridge University Press.

Davison, Joe, Joshua Feldman, and Alexander M Rush (2019). “Commonsense knowledge mining from pretrained models”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1173–1178.

Daza, Daniel and Michael Cochez (2020). “Message Passing for Query Answering over Knowledge Graphs”. In: *CoRR* abs/2002.02406. arXiv: 2002.02406. URL: https://arxiv.org/abs/2002.02406.

De Cao, Nicola, Gautier Izacard, Sebastian Riedel, and Fabio Petroni (2021). “Autoregressive Entity Retrieval”. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=5k8F6UU39V.

Dehghani, Mostafa, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser (2019). “Universal Transformers”. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=HyzdRiR9Y7.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In:

Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/n19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

Du, Nan, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P. Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen S. Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui (2022). "GLaM: Efficient Scaling of Language Models with Mixture-of-Experts". In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 5547–5569. URL: https://proceedings.mlr.press/v162/du22c.html.

Ellis, Kevin, Catherine Wong, Maxwell I. Nye, Mathias Sablé-Meyer, Luc Cary, Lucas Morales, Luke B. Hewitt, Armando Solar-Lezama, and Joshua B. Tenenbaum (2020). "DreamCoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning". In: *CoRR* abs/2006.08381. arXiv: 2006.08381. URL: https://arxiv.org/abs/2006.08381.

Fang, Zheng, Yanan Cao, Qian Li, Dongjie Zhang, Zhenyu Zhang, and Yanbing Liu (2019). "Joint Entity Linking with Deep Reinforcement Learning". In: *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. Ed. by Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia. ACM, pp. 438–447. DOI: 10.1145/3308558.3313517. URL: https://doi.org/10.1145/3308558.3313517.

Fedus, William, Barret Zoph, and Noam Shazeer (2022). "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity". In: *J. Mach. Learn. Res.* 23, 120:1–120:39. URL: http://jmlr.org/papers/v23/21-0998.html.

Feigenbaum, Edward A. and Julian Feldman (1963). *Computers and Thought*. USA: McGraw-Hill, Inc. ISBN: 0070203709.

Ferragina, Paolo and Ugo Scaiella (2010). "TAGME: on-the-fly annotation of short text fragments (by wikipedia entities)". In: *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*. Ed. by Jimmy X. Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An. ACM, pp. 1625–1628. DOI: 10.1145/1871437.1871689. URL: https://doi.org/10.1145/1871437.1871689.

Ferrucci, David, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Christopher Welty (2010). "Building Watson: An Overview of the DeepQA Project". In: *AI Magazine*.

Furrer, Daniel, Marc van Zee, Nathan Scales, and Nathanael Schärli (2020). "Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures". In: *CoRR* abs/2007.08970. arXiv: 2007.08970. URL: https://arxiv.org/abs/2007.08970.

Gabrilovich, Evgeniy, Michael Ringgaard, and Amarnag Subramanya (June 2013). "FACC1: Freebase annotation of ClueWeb corpora". In.

Ganapini, Marianna Bergamaschi, Murray Campbell, Francesco Fabiano, Lior Horesh, Jon Lenchner, Andrea Loreggia, Nicholas Mattei, Francesca Rossi, Biplav Srivastava, and Kristen Brent Venable (2022). "Thinking Fast and Slow in AI: The Role of Metacognition". In: *Machine Learning, Optimization, and Data Science - 8th International Workshop, LOD 2022, Certosa di Pontignano, Italy, September 19-22, 2022, Revised Selected Papers, Part II*. Ed. by Giuseppe Nicosia, Varun Ojha, Emanuele La Malfa, Gabriele La Malfa, Panos M. Pardalos, Giuseppe Di Fatta, Giovanni Giuffrida, and Renato Umeton. Vol. 13811. Lecture Notes in Computer Science. Springer, pp. 502–509. DOI: 10.1007/978-3-031-25891-6_38. URL: https://doi.org/10.1007/978-3-031-25891-6%5C_38.

Ganea, Octavian-Eugen and Thomas Hofmann (Sept. 2017). "Deep Joint Entity Disambiguation with Local Neural Attention". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 2619–2629. DOI: 10.18653/v1/D17-1277. URL: https://aclanthology.org/D17-1277.

Garnelo, Marta, Kai Arulkumaran, and Murray Shanahan (2016). "Towards deep symbolic reinforcement learning". In: *arXiv preprint arXiv:1609.05518*.

Gilmer, Justin, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl (2017). "Neural Message Passing for Quantum Chemistry". In: *34th International Conference on Machine Learning*. Vol. 70. ICML'17, pp. 1263–1272.

Gori, M., G. Monfardini, and F. Scarselli (2005). "A new model for learning in graph domains". In: *Proceedings of the IEEE International Joint Conference on Neural Networks*. Vol. 2, pp. 729–734.

Goyal, Anirudh, Aniket Rajiv Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Curtis Mozer, and Yoshua Bengio (2022). "Coordination Among Neural Modules Through a Shared Global Workspace". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. URL: https://openreview.net/forum?id=XzTtHjgPDsT.

Goyal, Yash, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh (2017). "Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, pp. 6325–6334. DOI: 10.1109/CVPR.2017.670. URL: https://doi.org/10.1109/CVPR.2017.670.

Graves, Alex, Greg Wayne, and Ivo Danihelka (2014). "Neural turing machines". In: *arXiv preprint arXiv:1410.5401*.

Graves, Alex, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, AdriàPuigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis (2016). "Hybrid computing using a neural network with dynamic external memory". In: *Nature* 538.

Gul, Muhammad Shahzeb Khan, Michel Bätz, and Joachim Keinert (2019). "Pixel-Wise Confidences for Stereo Disparities Using Recurrent Neural Networks". In: *30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019*. BMVA Press, p. 23. URL: https://bmvc2019.org/wp-content/uploads/papers/0274-paper.pdf.

Guo, Daya, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin (2018). "Dialog-to-Action: Conversational Question Answering Over a Large-Scale Knowledge Base". In: *Advances in Neural Information Processing Systems 31, NeurIPS 2018*, pp. 2946–2955.

Guo, Jiaqi, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang (2019). "Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*. Association for Computational Linguistics, pp. 4524–4535.

Guo, Xiaoxiao, Mo Yu, Yupeng Gao, Chuang Gan, Murray Campbell, and Shiyu Chang (Nov. 2020). "Interactive Fiction Game Playing as Multi-Paragraph Reading Comprehension with Reinforcement Learning". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 7755–7765. DOI: 10.18653/v1/2020.emnlp-main.624. URL: https://aclanthology.org/2020.emnlp-main.624.

Guo, Yinuo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang (2020). "Hierarchical Poset Decoding for Compositional Generalization in Language". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*.

Guu, Kelvin, John Miller, and Percy Liang (2015). "Traversing Knowledge Graphs in Vector Space". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*. The Association for Computational Linguistics, pp. 318–327.

Hadsell, Raia, Sumit Chopra, and Yann LeCun (2006). "Dimensionality Reduction by Learning an Invariant Mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*. IEEE Computer Society, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100. URL: https://doi.org/10.1109/CVPR.2006.100.

Hamilton, William L., Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec (2018). "Embedding Logical Queries on Knowledge Graphs". In: *NeurIPS 2018*.

Han, Xu, Shulin Cao, Lv Xin, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li (2018). “OpenKE: An Open Toolkit for Knowledge Embedding”. In: *Proceedings of EMNLP*.

Hausknecht, Matthew, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan (2020). “Interactive fiction games: A colossal adventure”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05, pp. 7903–7910.

He, Ji, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf (2016). “Deep Reinforcement Learning with a Natural Language Action Space”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics. DOI: 10.18653/v1/p16-1153. URL: https://doi.org/10.18653/v1/p16-1153.

He, Zhengyan, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang (2013). “Learning Entity Representation for Entity Disambiguation”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*. The Association for Computer Linguistics, pp. 30–34. URL: https://aclanthology.org/P13-2006/.

Higgins, Irina, Nicolas Sonnerat, Loic Matthey, Arka Pal, Christopher P. Burgess, Matko Bosnjak, Murray Shanahan, Matthew M. Botvinick, Demis Hassabis, and Alexander Lerchner (2018). “SCAN: Learning Hierarchical Compositional Visual Concepts”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=rkN2Il-RZ.

Hoang, Lan, Shivam Ratnakar, Nicolas Galichet, Akifumi Wachi, Keerthiram Murugesan, Songtao Lu, Mattia Atzeni, Michael Katz, and Subhajit Chaudhury (2022). “SCERL: A Benchmark for intersecting language and safe reinforcement learning”. In: *Second Workshop on Language and Reinforcement Learning, at NeurIPS 2022*. URL: https://openreview.net/forum?id=rNmrhsewsUX.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

Hoffart, Johannes, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum (2011). “Robust Disambiguation of Named Entities in Text”. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, pp. 782–792. URL: https://aclanthology.org/D11-1072/.

Hudson, Drew A. and Christopher D. Manning (2018). “Compositional Attention Networks for Machine Reasoning”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Hudson, Drew A. and Christopher D. Manning (2019). “Learning by Abstraction: The Neural State Machine”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada.* Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 5901–5914.

Jiang, Wei, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi (2021). “COTR: Correspondence Transformer for Matching Across Images”. In: *2021 International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021.* IEEE, pp. 6187–6197. DOI: 10.1109/ICCV48922.2021.00615. URL: https://doi.org/10.1109/ICCV48922.2021.00615.

Johnson, Jeff, Matthijs Douze, and Hervé Jégou (2021). “Billion-Scale Similarity Search with GPUs”. In: *IEEE Trans. Big Data* 7.3, pp. 535–547. DOI: 10.1109/TBDATA.2019.2921572. URL: https://doi.org/10.1109/TBDATA.2019.2921572.

Johnson, Justin, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick (2017). “CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017.* IEEE Computer Society, pp. 1988–1997. DOI: 10.1109/CVPR.2017.215. URL: https://doi.org/10.1109/CVPR.2017.215.

Johnson, Justin, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick (2017). “Inferring and Executing Programs for Visual Reasoning”. In: *International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017.* IEEE Computer Society, pp. 3008–3017. DOI: 10.1109/ICCV.2017.325. URL: https://doi.org/10.1109/ICCV.2017.325.

Juba, Brendan (2016). “Integrated common sense learning and planning in POMDPs”. In: *The Journal of Machine Learning Research* 17.1, pp. 3276–3312.

Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. (2021). “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873, pp. 583–589.

Kaelbling, Leslie Pack, Michael L Littman, and Anthony R Cassandra (1998). “Planning and acting in partially observable stochastic domains”. In: *Artificial intelligence* 101.1-2, pp. 99–134.

Kahneman, Daniel (2011). *Thinking, fast and slow.* New York: Farrar, Straus and Giroux. URL: https://www.amazon.de/Thinking-Fast-Slow-Daniel-Kahneman/dp/0374275637/ref=wl_it_dp_o_pdT1_nS_nC?ie=UTF8&colid=151193SNGKJT9&coliid=I3OCESLZCVDFL7.

Kapanipathi, Pavan, Veronika Thost, Siva Sankalp Patel, Spencer Whitehead, Ibrahim Abdelaziz, Avinash Balakrishnan, Maria Chang, Kshitij Fadnis, Chulaka Gunasekara, Bassem Makni, Nicholas Mattei, Kartik Talamadupula, and Achille Fokoue (2020).

"Infusing Knowledge into the Textual Entailment Task Using Graph Convolutional Networks". In: *AAAI*.

Kassner, Nora, Benno Krojer, and Hinrich Schütze (2020). "Are Pretrained Language Models Symbolic Reasoners over Knowledge?" In: *Proceedings of the 24th Conference on Computational Natural Language Learning, CoNLL 2020, Online, November 19-20, 2020*. Ed. by Raquel Fernández and Tal Linzen. Association for Computational Linguistics, pp. 552–564. DOI: 10.18653/V1/2020.CONLL-1.45. URL: https://doi.org/10.18653/v1/2020.conll-1.45.

Kassner, Nora and Hinrich Schütze (2020). "Negated and Misprimed Probes for Pre-trained Language Models: Birds Can Talk, But Cannot Fly". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*. Association for Computational Linguistics, pp. 7811–7818.

Keysers, Daniel, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet (2020). "Measuring Compositional Generalization: A Comprehensive Method on Realistic Data". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=SygcCnNKwr.

Khalife, Sammy and Michalis Vazirgiannis (2019). "Scalable graph-based method for individual named entity identification". In: *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing, TextGraphs@EMNLP 2019, Hong Kong, November 4, 2019*. Ed. by Dmitry Ustalov, Swapna Somasundaran, Peter Jansen, Goran Glavas, Martin Riedl, Mihai Surdeanu, and Michalis Vazirgiannis. Association for Computational Linguistics, pp. 17–25. DOI: 10.18653/v1/D19-5303. URL: https://doi.org/10.18653/v1/D19-5303.

Khalil, Elias B., Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song (2017). "Learning Combinatorial Optimization Algorithms over Graphs". In: *Advances in Neural Information Processing Systems 30, NeurIPS 2017*, pp. 6351–6361.

Kipf, Thomas N. and Max Welling (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net.

Klein, Tassilo and Moin Nabi (2019). "Attention is (not) all you need for commonsense reasoning". In: *arXiv preprint arXiv:1905.13497*.

Kolev, Victor, Bogdan Georgiev, and Svetlin Penkov (2020). "Neural Abstract Reasoner". In: *CoRR* abs/2011.09860. arXiv: 2011.09860. URL: https://arxiv.org/abs/2011.09860.

Kolitsas, Nikolaos, Octavian-Eugen Ganea, and Thomas Hofmann (Oct. 2018). "End-to-End Neural Entity Linking". In: *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, pp. 519–529. DOI: 10.18653/v1/K18-1050. URL: https://aclanthology.org/K18-1050.

Kolodner, Janet L (1983). “Reconstructive memory: A computer model”. In: *Cognitive science* 7.4, pp. 281–328.

Kool, Wouter, Herke van Hoof, and Max Welling (2019). “Attention, Learn to Solve Routing Problems!” In: *International Conference on Learning Representations, ICLR 2019*.

Krippendorff, Klaus (2018). *Content analysis: An introduction to its methodology*. Sage publications.

Krivosheev, Evgeny, Mattia Atzeni, Katsiaryna Mirylenka, Paolo Scotton, and Fabio Casati (2021). “Siamese Graph Neural Networks for Data Integration”. In: *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

Krivosheev, Evgeny, Mattia Atzeni, Katsiaryna Mirylenka, Paolo Scotton, Christoph Miksovic, and Anton Zorin (2021). “Business Entity Matching with Siamese Graph Convolutional Networks”. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, pp. 16054–16056. DOI: 10.1609/AAAI.V35I18.18010. URL: https://doi.org/10.1609/aaai.v35i18.18010.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. Ed. by Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, pp. 1106–1114. URL: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

Kumar, Ankit, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher (2016). “Ask Me Anything: Dynamic Memory Networks for Natural Language Processing”. In: *ICML 2016*.

Lai, Alice and Julia Hockenmaier (2017). “Learning to Predict Denotational Probabilities For Modeling Entailment”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. Ed. by Mirella Lapata, Phil Blunsom, and Alexander Koller. Association for Computational Linguistics, pp. 721–730. DOI: 10.18653/v1/e17-1068. URL: https://doi.org/10.18653/v1/e17-1068.

Lake, Brenden M. and Marco Baroni (2018). “Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks”. In: *ICML*.

Lake, Brenden M., Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman (2017). “Building machines that learn and think like people”. In: *Behavioral and Brain Sciences* 40, e253. DOI: 10.1017/S0140525X16001837.

Le, Phong and Ivan Titov (July 2019). “Boosting Entity Linking Performance by Leveraging Unlabeled Documents”. In: *Proceedings of the 57th Annual Meeting of the*

Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, pp. 1935–1945. DOI: 10.18653/v1/P19-1187. URL: https://aclanthology.org/P19-1187.

LeCun, Yann (2022). "A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27". In.

LeCun, Yann, Yoshua Bengio, and Geoffrey E. Hinton (2015). "Deep learning". In: *Nat.* 521.7553, pp. 436–444. DOI: 10.1038/NATURE14539. URL: https://doi.org/10.1038/nature14539.

Li, Xiang, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum (2019). "Smoothing the Geometry of Probabilistic Box Embeddings". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=H1xSNiRcF7.

Li, Yujia, Felix Gimeno, Pushmeet Kohli, and Oriol Vinyals (2020). "Strong Generalization and Efficiency in Neural Programs". In: *CoRR* abs/2007.03629. URL: https://arxiv.org/abs/2007.03629.

Lin, Bill Yuchen, Xinyue Chen, Jamin Chen, and Xiang Ren (2019). "KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP*. Association for Computational Linguistics, pp. 2829–2839.

Lin, Long-Ji (1992). "Self-improving reactive agents based on reinforcement learning, planning and teaching". In: *Machine learning* 8.3-4, pp. 293–321.

– (1993). *Reinforcement learning for robots using neural networks*. Tech. rep. Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

Lin, Yankai, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu (2015). "Modeling Relation Paths for Representation Learning of Knowledge Bases". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*. The Association for Computational Linguistics, pp. 705–714.

Lindblad, Joakim and Natasa Sladoje (2014). "Linear Time Distances Between Fuzzy Sets With Applications to Pattern Matching and Classification". In: *IEEE Trans. Image Process.* 23.1, pp. 126–136. DOI: 10.1109/TIP.2013.2286904. URL: https://doi.org/10.1109/TIP.2013.2286904.

Lippi, Marco (2016). "Reasoning with Deep Learning: an Open Challenge". In: *Proceedings of the AI*IA Workshop on Deep Understanding and Reasoning: A Challenge for Next-generation Intelligent Agents*.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692. arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

Locatello, Francesco, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem (2019). "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations”. In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 4114–4124. URL: http://proceedings.mlr.press/v97/locatello19a.html.

Loshchilov, Ilya and Frank Hutter (2019). “Decoupled Weight Decay Regularization”. In: *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net.

Loukas, Andreas (2019). “Graph Reduction with Spectral and Cut Guarantees”. In: *J. Mach. Learn. Res.* 20, 116:1–116:42. URL: http://jmlr.org/papers/v20/18-680.html.

Loukas, Andreas and Pierre Vandergheynst (2018). “Spectrally Approximating Large Graphs with Smaller Graphs”. In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3243–3252. URL: http://proceedings.mlr.press/v80/loukas18a.html.

Lowe, D.G. (1999). “Object recognition from local scale-invariant features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

Lu, Jiahao, Johan Öfverstedt, Joakim Lindblad, and Natasa Sladoje (2021). “Is Image-to-Image Translation the Panacea for Multimodal Image Registration? A Comparative Study”. In: *CoRR* abs/2103.16262. arXiv: 2103.16262. URL: https://arxiv.org/abs/2103.16262.

Luong, Thang, Hieu Pham, and Christopher D. Manning (Sept. 2015). “Effective Approaches to Attention-based Neural Machine Translation”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://www.aclweb.org/anthology/D15-1166.

Maaten, Laurens van der and Geoffrey Hinton (2008). “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9, pp. 2579–2605. URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Mao, Jiayuan, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu (2019). “The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision”. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. URL: https://openreview.net/forum?id=rJgMlhRctm.

Marcus, Gary (2018). “Deep Learning: A Critical Appraisal”. In: *CoRR* abs/1801.00631. arXiv: 1801.00631.

– (2020). *The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence*. arXiv: 2002.06177 [cs.AI].

Martin, Lara J., Srijan Sood, and Mark Riedl (2018). “Dungeons and DQNs: Toward Reinforcement Learning Agents that Play Tabletop Roleplaying Games”. In: *Pro-*

ceedings of the Joint Workshop on Intelligent Narrative Technologies and Workshop on Intelligent Cinematography and Editing co-located with 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, INTWICED@AIIDE 2018, Edmonton, Alberta, Canada, November 13-14, 2018. Ed. by Hui-Yin Wu, Mei Si, and Arnav Jhala. Vol. 2321. CEUR Workshop Proceedings. CEUR-WS.org. URL: http://ceur-ws.org/Vol-2321/paper4.pdf.

Martinkus, Karolis, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer (2022). "SPECTRE: Spectral Conditioning Helps to Overcome the Expressivity Limits of One-shot Graph Generators". In: *International Conference on Machine Learning*. ICML.

Mikolov, Tomás, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, pp. 3111–3119. URL: https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

Miller, Alexander H., Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston (2016). "Key-Value Memory Networks for Directly Reading Documents". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*. The Association for Computational Linguistics, pp. 1400–1409.

Milne, David N. and Ian H. Witten (2008). "Learning to link with wikipedia". In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*. Ed. by James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury. ACM, pp. 509–518. DOI: 10.1145/1458082.1458150. URL: https://doi.org/10.1145/1458082.1458150.

Mnih, Volodymyr, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu (2016). "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*, pp. 1928–1937.

Muggleton, Stephen H. and Luc De Raedt (1994). "Inductive Logic Programming: Theory and Methods". In: *J. Log. Program.* 19/20, pp. 629–679. DOI: 10.1016/0743-1066(94)90035-3. URL: https://doi.org/10.1016/0743-1066(94)90035-3.

Mulang, Isaiah Onando, Kuldeep Singh, Chaitali Prabhu, Abhishek Nadgeri, Johannes Hoffart, and Jens Lehmann (2020). "Evaluating the Impact of Knowledge Graph Context on Entity Disambiguation Models". In: *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. CIKM '20. Virtual Event, Ireland: Association for Computing Machinery, pp. 2157–2160. ISBN: 9781450368599. DOI: 10.1145/3340531.3412159. URL: https://doi.org/10.1145/3340531.3412159.

Murugesan, K., Subhajit Chaudhury, and Kartik Talamadupula (2021). “Eye of the Beholder: Improved Relation Generalization for Text-based Reinforcement Learning Agents”. In: *ArXiv* abs/2106.05387.

Murugesan, Keerthiram, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell (2021). “Text-based RL Agents with Commonsense Knowledge: New Challenges, Environments and Baselines”. In: *Thirty Fifth AAAI Conference on Artificial Intelligence*.

Murugesan, Keerthiram, Mattia Atzeni, Pavan Kapanipathi, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell (2021). “Efficient Text-based Reinforcement Learning by Jointly Leveraging State and Commonsense Graph Representations”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 719–725.

Narasimhan, Karthik, Tejas Kulkarni, and Regina Barzilay (2015). “Language understanding for text-based games using deep reinforcement learning”. In: *arXiv preprint arXiv:1506.08941*.

Onoe, Yasumasa and Greg Durrett (2020). “Fine-Grained Entity Typing for Domain Independent Entity Linking”. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, pp. 8576–8583. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6380.

OpenAI (2023). “GPT-4 Technical Report”. In: *CoRR* abs/2303.08774. DOI: 10.48550/ARXIV.2303.08774. arXiv: 2303.08774. URL: https://doi.org/10.48550/arXiv.2303.08774.

Orr, Laurel J., Megan Leszczynski, Neel Guha, Sen Wu, Simran Arora, Xiao Ling, and Christopher Ré (2021). “Bootleg: Chasing the Tail with Self-Supervised Named Entity Disambiguation”. In: *11th Conference on Innovative Data Systems Research, CIDR 2021, Virtual Event, January 11-15, 2021, Online Proceedings*. www.cidrdb.org. URL: http://cidrdb.org/cidr2021/papers/cidr2021%5C_paper13.pdf.

Otto, Samuel E., Nicholas Zolman, J. Nathan Kutz, and Steven L. Brunton (2023). “A Unified Framework to Enforce, Discover, and Promote Symmetry in Machine Learning”. In: *CoRR* abs/2311.00212. DOI: 10.48550/ARXIV.2311.00212. arXiv: 2311.00212. URL: https://doi.org/10.48550/arXiv.2311.00212.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). “Glove: Global Vectors for Word Representation”. In: *EMNLP*.

Pershina, Maria, Yifan He, and Ralph Grishman (2015). “Personalized Page Rank for Named Entity Disambiguation”. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, pp. 238–243. DOI: 10.3115/v1/N15-1026. URL: https://aclanthology.org/N15-1026.

Petroni, Fabio, Aleksandra Piktus, Angela Fan, Patrick S. H. Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel (2021). "KILT: a Benchmark for Knowledge Intensive Language Tasks". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*. Ed. by Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou. Association for Computational Linguistics, pp. 2523–2544. DOI: 10.18653/v1/2021.naacl-main.200. URL: https://doi.org/10.18653/v1/2021.naacl-main.200.

Pielawski, Nicolas, Elisabeth Wetzer, Johan Öfverstedt, Jiahao Lu, Carolina Wählby, Joakim Lindblad, and Natasa Sladoje (2020). "CoMIR: Contrastive Multimodal Image Representation for Registration". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin.

Prabhudesai, Mihir, Shamit Lal, Darshan Patil, Hsiao-Yu Tung, Adam W. Harley, and Katerina Fragkiadaki (2021). "Disentangling 3D Prototypical Networks for Few-Shot Concept Learning". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=-Lr-u0b42he.

Radford, Alec, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). *Language Models are Unsupervised Multitask Learners*.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *J. Mach. Learn. Res.* 21, 140:1–140:67.

Raiman, Jonathan and Olivier Raiman (2018). "DeepType: Multilingual Entity Linking by Neural Type System Evolution". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, pp. 5406–5413. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17148.

Ratinov, Lev-Arie, Dan Roth, Doug Downey, and Mike Anderson (2011). "Local and Global Algorithms for Disambiguation to Wikipedia". In: *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*. Ed. by Dekang Lin, Yuji Matsumoto, and Rada Mihalcea. The Association for Computer Linguistics, pp. 1375–1384. URL: https://aclanthology.org/P11-1138/.

Reed, Scott and Nando De Freitas (2015). "Neural programmer-interpreters". In: *arXiv preprint arXiv:1511.06279*.

Ren, Hongyu, Weihua Hu, and Jure Leskovec (2020). “Query2box: Reasoning over Knowledge Graphs in Vector Space Using Box Embeddings”. In: *ICLR*.

Ren, Hongyu and Jure Leskovec (2020). “Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020.*

Sachan, Mrinmaya (2020). “Knowledge Graph Embedding Compression”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020.* Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault. Association for Computational Linguistics, pp. 2681–2691. DOI: 10.18653/v1/2020.acl-main.238. URL: https://doi.org/10.18653/v1/2020.acl-main.238.

Sagawa, Shiori, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang (2019). “Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization”. In: *CoRR* abs/1911.08731. arXiv: 1911.08731. URL: http://arxiv.org/abs/1911.08731.

Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf (2019). “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *CoRR* abs/1910.01108. arXiv: 1910.01108. URL: http://arxiv.org/abs/1910.01108.

Sap, Maarten, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi (2019). “Atomic: An atlas of machine commonsense for if-then reasoning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 33, pp. 3027–3035.

Sarlin, Paul-Edouard, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich (2020). “SuperGlue: Learning Feature Matching With Graph Neural Networks”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020.* Computer Vision Foundation / IEEE, pp. 4937–4946. DOI: 10.1109/CVPR42600.2020.00499. URL: https://openaccess.thecvf.com/content%5C_CVPR%5C_2020/html/Sarlin%5C_SuperGlue%5C_Learning%5C_Feature%5C_Matching%5C_With%5C_Graph%5C_Neural%5C_Networks%5C_CVPR%5C_2020%5C_paper.html.

Sartran, Laurent, Samuel Barrett, Adhiguna Kuncoro, Milos Stanojevic, Phil Blunsom, and Chris Dyer (2022). “Transformer Grammars: Augmenting Transformer Language Models with Syntactic Inductive Biases at Scale”. In: *Trans. Assoc. Comput. Linguistics* 10, pp. 1423–1439. URL: https://transacl.org/ojs/index.php/tacl/article/view/3951.

Saxena, Apoorv, Aditay Tripathi, and Partha P. Talukdar (2020). “Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020.* Association for Computational Linguistics, pp. 4498–4507.

Scarselli, Franco, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini (2009). “The Graph Neural Network Model”. In: *IEEE Trans. Neural Networks* 20.1, pp. 61–80.

Schank, Roger C (1983). *Dynamic memory: A theory of reminding and learning in computers and people.* cambridge university press.

Schölkopf, Bernhard, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio (2021). "Toward Causal Representation Learning". In: *Proc. IEEE* 109.5, pp. 612–634. DOI: 10.1109/JPROC.2021.3058954. URL: https://doi.org/10.1109/JPROC.2021.3058954.

Selsam, Daniel, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill (2018). "Learning a SAT Solver from Single-Bit Supervision". In: *CoRR* abs/1802.03685. arXiv: 1802.03685.

Seo, Min Joon, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi (2017). "Bidirectional Attention Flow for Machine Comprehension". In: *5th International Conference on Learning Representations, ICLR 2017.*

Shahbazi, Hamed, Xiaoli Z. Fern, Reza Ghaeini, Rasha Obeidat, and Prasad Tadepalli (2019). "Entity-aware ELMo: Learning Contextual Entity Representation for Entity Disambiguation". In: *CoRR* abs/1908.05762. arXiv: 1908.05762. URL: http://arxiv.org/abs/1908.05762.

Shaw, Peter, Jakob Uszkoreit, and Ashish Vaswani (2018). "Self-Attention with Relative Position Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers).* Ed. by Marilyn A. Walker, Heng Ji, and Amanda Stent. Association for Computational Linguistics, pp. 464–468. DOI: 10.18653/v1/n18-2074. URL: https://doi.org/10.18653/v1/n18-2074.

Singh, Ishika, Gargi Singh, and Ashutosh Modi (2021). "Pre-trained Language Models as Prior Knowledge for Playing Text-based Games". In: *CoRR* abs/2107.08408. arXiv: 2107.08408. URL: https://arxiv.org/abs/2107.08408.

Snell, Jake, Kevin Swersky, and Richard S. Zemel (2017). "Prototypical Networks for Few-shot Learning". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA.* Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 4077–4087. URL: https://proceedings.neurips.cc/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html.

So, David R., Quoc V. Le, and Chen Liang (2019). "The Evolved Transformer". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019.* Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 5877–5886.

Spaan, Matthijs TJ (2012). "Partially observable Markov decision processes". In: *Reinforcement Learning.* Springer, pp. 387–414.

Speer, Robert, Joshua Chin, and Catherine Havasi (2017). "ConceptNet 5.5: An Open Multilingual Graph of General Knowledge." In: *AAAI*, pp. 4444–4451.

Spelke, Elizabeth S. and Katherine D. Kinzler (2007). "Core knowledge". In: *Developmental Science* 10.1, pp. 89–96. DOI: https://doi.org/10.1111/j.1467-7687.2007.00569.x.

eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-7687.2007.00569.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-7687.2007.00569.x.

Stärk, Hannes, Octavian-Eugen Ganea, Lagnajit Pattanaik, Regina Barzilay, and Tommi Jaakkola (2022). *EquiBind: Geometric Deep Learning for Drug Binding Structure Prediction.*

Sukhbaatar, Sainbayar, Arthur Szlam, Jason Weston, and Rob Fergus (2015). "End-To-End Memory Networks". In: *Advances in Neural Information Processing Systems 28*, pp. 2440–2448.

Sun, Haitian, Andrew O. Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W. Cohen (2020). "Faithful Embeddings for Knowledge Base Queries". In: *NeurIPS 2020*.

Sun, Haitian, Tania Bedrax-Weiss, and William W. Cohen (2019). "PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text". In: *EMNLP*.

Sun, Haitian, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen (2018). "Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text". In: *EMNLP*.

Sun, Yaming, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang (2015). "Modeling Mention, Context and Entity with Neural Networks for Entity Disambiguation". In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. Ed. by Qiang Yang and Michael J. Wooldridge. AAAI Press, pp. 1333–1339. URL: http://ijcai.org/Abstract/15/192.

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement Learning: An Introduction.* Cambridge, MA, USA: A Bradford Book. ISBN: 0262039249.

Sverrisson, Freyr, Jean Feydy, Joshua Southern, Michael M Bronstein, and Bruno Correia (2022). "Physics-informed deep neural network for rigid-body protein docking". In: *ICLR2022 Machine Learning for Drug Discovery*.

Talmor, Alon, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant (2018). "Commonsenseqa: A question answering challenge targeting commonsense knowledge". In: *arXiv preprint arXiv:1811.00937*.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30, NeurIPS 2017*, pp. 6000–6010.

Velickovic, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio (2018). "Graph Attention Networks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.* OpenReview.net. URL: https://openreview.net/forum?id=rJXMpikCZ.

Vendrov, Ivan, Ryan Kiros, Sanja Fidler, and Raquel Urtasun (2016). "Order-Embeddings of Images and Language". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings.* Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1511.06361.

Vilnis, Luke, Xiang Li, Shikhar Murty, and Andrew McCallum (2018). “Probabilistic Embedding of Knowledge Graphs with Box Lattice Measures”. In: *56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. Ed. by Iryna Gurevych and Yusuke Miyao. Association for Computational Linguistics, pp. 263–272. DOI: 10.18653/v1/P18-1025. URL: https://aclanthology.org/P18-1025/.

Vinyals, Oriol, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra (2016). “Matching Networks for One Shot Learning”. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, pp. 3630–3638. URL: https://proceedings.neurips.cc/paper/2016/hash/90e1357833654983612fb05e3ec9148c-Abstract.html.

Vrandečić, Denny and Markus Krötzsch (2014). “Wikidata: A Free Collaborative Knowledgebase”. In: *Commun. ACM* 57.10, pp. 78–85. ISSN: 0001-0782. DOI: 10.1145/2629489. URL: https://doi.org/10.1145/2629489.

Wang, Quan, Zhendong Mao, Bin Wang, and Li Guo (2017). “Knowledge Graph Embedding: A Survey of Approaches and Applications”. In: *IEEE Trans. Knowl. Data Eng.* 29.12, pp. 2724–2743.

Wang, Tan, Zhongqi Yue, Jianqiang Huang, Qianru Sun, and Hanwang Zhang (2021). “Self-Supervised Learning Disentangled Group Representation as Feature”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc’Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, pp. 18225–18240. URL: https://proceedings.neurips.cc/paper/2021/hash/97416ac0f58056947e2eb5d5d253d4f2-Abstract.html.

Wang, Xiaolong, Ross B. Girshick, Abhinav Gupta, and Kaiming He (2018). “Non-Local Neural Networks”. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*. IEEE Computer Society, pp. 7794–7803.

Wang, Ziyu, Victor Bapst, Nicolas Heess, Volodymyr Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas (2017). “Sample Efficient Actor-Critic with Experience Replay”. In: *5th International Conference on Learning Representations, ICLR 2017*.

Weston, Jason, Sumit Chopra, and Antoine Bordes (2015). “Memory Networks”. In: *3rd International Conference on Learning Representations, ICLR 2015*.

Wind, Johan Sokrates (2020). *DSL solution to the ARC challenge*. URL: https://github.com/top-quarks/ARC-solution/blob/master/ARC-solution_documentation.pdf.

Wu, Ledell, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer (2020). “Scalable Zero-shot Entity Linking with Dense Entity Retrieval”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Association for Computational Linguistics, pp. 6397–6407. DOI:

10.18653/v1/2020.emnlp-main.519. URL: https://doi.org/10.18653/v1/2020.emnlp-main.519.

Xiong, Wenhan, Thien Hoang, and William Yang Wang (2017). “DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning”. In: *EMNLP*.

Xu, Keyulu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka (2020). “What Can Neural Networks Reason About?” In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=rJxbJeHFPS.

Xu, Yunqiu, Ling Chen, Meng Fang, Yang Wang, and Chengqi Zhang (2020). “Deep Reinforcement Learning with Transformers for Text Adventure Games”. In: *IEEE Conference on Games, CoG 2020, Osaka, Japan, August 24-27, 2020*. IEEE, pp. 65–72. DOI: 10.1109/CoG47356.2020.9231622. URL: https://doi.org/10.1109/CoG47356.2020.9231622.

Yamada, Ikuya, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji (2016). “Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation”. In: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*. Ed. by Yoav Goldberg and Stefan Riezler. ACL, pp. 250–259. DOI: 10.18653/v1/k16-1025. URL: https://doi.org/10.18653/v1/k16-1025.

Yan, Yujun, Kevin Swersky, Danai Koutra, Parthasarathy Ranganathan, and Milad Hashemi (2020). “Neural Execution Engines: Learning to Execute Subroutines”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin.

Yang, Mengyue, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang (2021). “CausalVAE: Disentangled Representation Learning via Neural Structural Causal Models”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, pp. 9593–9602. DOI: 10.1109/CVPR46437.2021.00947. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Yang%5C_CausalVAE%5C_Disentangled%5C_Representation%5C_Learning%5C_via%5C_Neural%5C_Structural%5C_Causal%5C_Models%5C_CVPR%5C_2021%5C_paper.html.

Yang, Xiyuan, Xiaotao Gu, Sheng Lin, Siliang Tang, Yueting Zhuang, Fei Wu, Zhigang Chen, Guoping Hu, and Xiang Ren (Nov. 2019). “Learning Dynamic Context Augmentation for Global Entity Linking”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 271–281. DOI: 10.18653/v1/D19-1026. URL: https://aclanthology.org/D19-1026.

Yang, Yi, Ozan Irsoy, and Kazi Shefaet Rahman (2018). “Collective Entity Disambiguation with Structured Gradient Tree Boosting”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. Ed. by Marilyn A. Walker, Heng Ji, and Amanda Stent. Association for Computational Linguistics, pp. 777–786. DOI: 10.18653/v1/n18-1071. URL: https://doi.org/10.18653/v1/n18-1071.

Yao, Xuchen and Benjamin Van Durme (2014). “Information Extraction over Structured Data: Question Answering with Freebase”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, Volume 1: Long Papers*. The Association for Computer Linguistics, pp. 956–966.

Yasunaga, Michihiro, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec (2021). “QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*. Association for Computational Linguistics, pp. 535–546. DOI: 10.18653/v1/2021.naacl-main.45.

Yih, Wen-tau, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao (2015). “Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base”. In: *Proceedings of the 53rd Annual Meeting of the ACL*.

Yu, Adams Wei, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le (2018). “Qanet: Combining local convolution with global self-attention for reading comprehension”. In: *arXiv preprint arXiv:1804.09541*.

Zahavy, Tom, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor (2018). “Learn what not to learn: Action elimination with deep reinforcement learning”. In: *Advances in Neural Information Processing Systems*, pp. 3562–3573.

Zhang, Yuyu, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song (2018). “Variational Reasoning for Question Answering With Knowledge Graph”. In: *AAAI*.

Zoph, Barret, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus (2022). “Designing Effective Sparse Expert Models”. In: *CoRR* abs/2202.08906. arXiv: 2202.08906. URL: https://arxiv.org/abs/2202.08906.

Mattia Atzeni

Email: mattia.atzeni@outlook.it
Google Scholar: https://scholar.google.it/citations?user=GxcjDq0AAAAJ

Education

EPFL – Swiss Federal Institute of Technology Lausanne — Lausanne, Switzerland
PhD in Machine Learning — *Jan 2020 – Dec 2023*
Advisors: Pierre Vandergheynst, Andreas Loukas

ETH Zürich – Swiss Federal Institute of Technology Zurich — Zurich, Switzerland
Special PhD student program, Academic guest — *Mar 2022 – Aug 2022*
Advisor: Mrinmaya Sachan

University of Cagliari — Cagliari, Italy
Master of Science in Computer Science – 110/110 *cum laude* — *Oct 2016 – Apr 2018*
Advisor: Maurizio Atzori, *GPA*: 29.9/30

University of Cagliari — Cagliari, Italy
Bachelor of Science in Computer Science – 110/110 *cum laude* — *Oct 2013 – Jul 2016*
Advisor: Maurizio Atzori, *GPA*: 29.8/30

Work Experience

EPFL – Swiss Federal Institute of Technology Lausanne — Lausanne, Switzerland
Doctoral Assistant — *Jan 2020 – Dec 2023*

- Research on methods for infusing prior structured knowledge in neural networks to improve their symbolic reasoning abilities and their sample efficiency.

Meta AI — London, United Kingdom
Research Scientist Intern — *Aug 2022 – Mar 2023*

- Explored methods for infusing prior knowledge in the latent space of entity linking models.
- Reached state-of-the-art results on standard entity-linking datasets and improved the performance of the model by up to 7.9 F1 points.

ETH Zürich – Institute for Machine Learning — Zurich, Switzerland
Visiting Researcher — *Apr 2022 – Aug 2022*

- Researched methods for improving the geometric reasoning abilities of neural networks.
- Published a variant of the transformer encoder which is 2 orders of magnitude more sample efficient and significantly exceeds state-of-the-art results on geometric-reasoning datasets.

IBM Research AI — Zurich, Switzerland
Research Scientist — *Mar 2019 – Feb 2022*

- Created datasets and algorithms for reinforcement learning in textual environments, achieving state-of-the-art results and leading to publications in top-tier conferences.
- Designed approaches to knowledge-based question answering, improving results on widely used datasets while increasing scalability by more than 2 orders of magnitude.

BUP solutions — Rome, Italy
Research Intern — *Apr 2017 – Oct 2017*

- Developed deep-learning approaches to sentiment analysis in the financial domain.
- Achieved a 10% improvement in accuracy compared to the model in production.

University of Cagliari — Cagliari, Italy
Research Assistant — *Mar 2016 – Mar 2019*

- Worked on semantic parsing and designed novel approaches to translating natural language utterances into source code.
- Reached state-of-the-art results and a 9% improvement compared to other commercial systems.

Professional Service and Volunteering

- *Program Committee member* for top-tier conferences in artificial intelligence, including NeurIPS (2021, 2022, 2023), ICML (2022, 2023), AAAI (2021, 2022, 2023).
- Serving as *Mentor* for *LeadTheFuture*, a Forbes-awarded organization that helps excellent students in STEM with one-to-one mentoring. Currently mentoring 6 master students of computer science.
- Supervised students at *ETH Zürich*: Shreya Sharma and Saüc Abadal Lloret.

Awards and Honors

- **Best Graduate Student of the School of Science**: awarded yearly by the University of Cagliari to one student who successfully graduated in the previous academic year. The school of science includes the departments of physics, chemistry, geology, mathematics and computer science.
- **First Ascent 2018**: selected nationwide among the 20 best Italian students of Computer Science. The selection process was based on academic achievements and practical tests on inductive and deductive reasoning, problem solving capabilities, and programming skills.
- **Top Theses of the University of Cagliari**: award released to the Bachelor's thesis: *CodeOntology: RDF-ization of Source Code*, which was the only awarded Bachelor's thesis in Computer Science.
- **Top Undergraduate Students of the University of Cagliari**: awarded €1000 from the University of Cagliari, based on the grade point average and the time taken to graduate.
- **Patent Application Award**: awarded by IBM for the patent applications filed in 2020.
- **Best Paper Award**: IEEE International Conference on Artificial Intelligence and Knowledge Engineering 2018, Laguna Hills, California, USA.
- **Best Demo Award**: Extended Semantic Web Conference 2018, Heraklion, Greece, 2018.
- **1st place** at the **ESWC 2017 Challenge on Semantic Sentiment Analysis**, June 2017.
- **INPS Scholarship**: awarded nationwide based on academic achievements (€10 000).
- **National Register of Excellent Students**: awarded nationwide as an excellent student by the Italian Ministry of Education, University and Research (MIUR), 2013.

Patents

1. Francesco Fusco, Mattia Atzeni, Abderrahim Labbi. *Bootstrapping of Text Classifiers*. Patent owner: IBM Research. US Patent Office, Filed, 2020.

Publications

Published 23 papers (upwards of 450 citations, h-index 11), including top-tier conferences like NeurIPS, ICLR, ICML, ACL, EMNLP, and AAAI. More details on Google Scholar.

1. Mattia Atzeni, Mikhail Plekhanov, Frédéric A. Dreyer, Nora Kassner, Simone Merello, Louis Martin, Nicola Cancedda. *Polar Ducks and Where to Find Them: Enhancing Entity Linking with Duck Typing and Polar Box Embeddings*, 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023.
2. Mattia Atzeni, Mrinmaya Sachan, Andreas Loukas. *Infusing Lattice Symmetry Priors in Attention Mechanisms for Sample-Efficient Abstract Geometric Reasoning*, International Conference on Machine Learning, ICML 2023.
3. Ngoc Lan Hoang, Nicolas Galichet, Akifumi Wachi, Shivam Ratnakar, Keerthiram Murugesan, Declan Millar, Songtao Lu, Mattia Atzeni, Michael Katz, Subhajit Chaudhury. *SCERL: A Benchmark for Intersecting Language and Safe Reinforcement Learning"*, Language and Reinforcement Learning Workshop (LaReL) at NeurIPS 2022.
4. Mattia Atzeni, Shehzaad Dhuliawala, Keerthiram Murugesan, and Mrinmaya Sachan. *Case-based Reasoning for Better Generalization in Textual Reinforcement Learning*, International Conference on Learning Representations, ICLR 2022.

5. Mattia Atzeni, Jasmina Bogojeska, and Andreas Loukas. *SQALER: Scaling Question Answering by Decoupling Multi-Hop and Logical Reasoning*, Advances in Neural Information Processing Systems, NeurIPS 2021.
6. Kinjal Basu, Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Kartik Talamadupula, Tim Klinger, Murray Campbell, Mrinmaya Sachan, Gopal Gupta. *A Hybrid Neuro-Symbolic approach for Text-Based Games using Inductive Logic Programming*, CLeaR workshop at AAAI 2022, Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations.
7. Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. *Efficient Text-based Reinforcement Learning by Jointly Leveraging State and Commonsense Graph Representations*, Proceedings of the 59^{th} Annual Meeting of the Association for Computational Linguistics, ACL 2021.
8. Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. *Text-based RL Agents with Commonsense Knowledge: New Challenges, Environments and Baselines*, The 35th AAAI Conference on Artificial Intelligence, AAAI 2021.
9. Evgeny Krivosheev, Mattia Atzeni, Katsiaryna Mirylenka, Paolo Scotton, Christoph Miksovic, and Anton Zorin. *Business Entity Matching with Siamese Graph Convolutional Networks*, The 35th AAAI Conference on Artificial Intelligence, AAAI 2021.
10. Evgeny Krivosheev, Mattia Atzeni, Katsiaryna Mirylenka, Paolo Scotton, and Fabio Casati. *Siamese Graph Convolutional Networks for Data Integration*, The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD 2021.
11. Mattia Atzeni and Diego Reforgiato Recupero. *Multi-Domain Sentiment Analysis with Mimicked and Polarized Word Embeddings for Human-Robot Interaction*, Future Generation Computer Systems, 2020.
12. Mattia Atzeni and Maurizio Atzori. *What is the cube root of 27? Question Answering over CodeOntology*, Proceedings of the International Semantic Web Conference, ISWC 2018 (**Spotlight**).
13. Mattia Atzeni and Maurizio Atzori. *AskCO: A Multi-Language and Extensible Smart Virtual Assistant*, IEEE International Conference on Artificial Intelligence and Knowledge Engineering, 2019.
14. Mattia Atzeni and Maurizio Atzori. *Translating Natural Language to Code: an Unsupervised Ontology-based Approach*, Proceedings of the IEEE International Conference on Artificial Intelligence and Knowledge Engineering, 2018 (**Best paper award**).
15. Mattia Atzeni and Diego Reforgiato Recupero. *Fine-Tuning of Word Embeddings for Semantic Sentiment Analysis*, Semantic Web Challenges - 5th SemWebEval Challenge at ESWC 2018.
16. Mattia Atzeni and Diego Reforgiato Recupero. *Deep Learning and Sentiment Analysis for Human-Robot Interaction*, ESWC 2018 Posters and Demonstrations Track, co-located with 15th Extended Semantic Web Conference (ESWC 2018), 2018 (**Best demo award**).
17. Amna Dridi, Mattia Atzeni, and Diego Reforgiato Recupero. *FineNews: fine-grained semantic sentiment analysis on financial microblogs and news*, International Journal of Machine Learning and Cybernetics, 2018.
18. Mattia Atzeni, Amna Dridi, and Diego Reforgiato Recupero. *Using Frame-Based Resources for Sentiment Analysis within the Financial Domain*, Progress in Artificial Intelligence, 2018.
19. Mattia Atzeni and Maurizio Atzori. *Towards Semantic Approaches for General-Purpose End-User Development*, 2018 Second IEEE International Conference on Robotic Computing (IRC), 2018.
20. Mattia Atzeni and Maurizio Atzori. *CodeOntology: RDF-ization of Source Code*, Proceedings of the International Semantic Web Conference, ISWC 2017 (**Spotlight**).
21. Mattia Atzeni and Maurizio Atzori. *CodeOntology: Querying Source Code in a Semantic Framework*, Proceedings of the ISWC 2017 Posters and Demonstrations Track.
22. Amna Dridi, Mattia Atzeni, and Diego Reforgiato Recupero. *Bearish-Bullish Sentiment Analysis on Financial Microblogs*, Proceedings of the 3rd International Workshop on Emotions, Modality, Sentiment Analysis and the Semantic Web, co-located with 14th ESWC 2017.
23. Mattia Atzeni, Amna Dridi, and Diego Reforgiato Recupero. *Fine-Grained Sentiment Analysis on Financial Microblogs and News Headlines*, Semantic Web Challenges - 4th SemWebEval Challenge at ESWC 2017.